NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

THE ROTATED SPEEDED-UP ROBUST FEATURES ALGORITHM (R-SURF)

by

Sean M. Jurgensen

June 2014

Thesis Advisor: Monique P. Fargues
Co-Advisor: Roberto Cristi

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2014	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE
THE ROTATED SPEEDED-UP ROBUST FEATURES ALGORITHM (R-SURF)

5. FUNDING NUMBERS

6. AUTHOR(S) Sean M. Jurgensen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943–5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

Weaknesses in the Fast Hessian detector utilized by the speeded-up robust features (SURF) algorithm are examined in this research. We evaluate the SURF algorithm to identify possible areas for improvement in the performance.

A proposed alternative to the SURF detector is proposed called rotated SURF (R-SURF). This method utilizes filters that are rotated 45 degrees counter-clockwise, and this modification is tested with standard detector testing methods against the regular SURF detector. Performance testing shows that the R-SURF outperforms the regular SURF detector when subject to image blurring, illumination changes and compression. Based on the testing results, the R-SURF detector outperforms regular SURF slightly when subjected to affine (viewpoint) changes. For image scale and rotation transformations, R-SURF outperforms for very small transformation values, but the regular SURF algorithm performs better for larger variations. The application of this research in the larger recognition process is also discussed.

14. SUBJECT TERMS SURF, speeded-up robust features, feature detector, feature detection, box filtering, box filters, Fast Hessian	**15. NUMBER OF PAGES** 135
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01–280–5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

THE ROTATED SPEEDED-UP ROBUST FEATURES ALGORITHM (R-SURF)

Sean M. Jurgensen
Lieutenant, United States Navy
B.S.E.E., University of Idaho, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2014

Author: Sean M. Jurgensen

Approved by: Dr. Monique P. Fargues
Thesis Advisor

Dr. Roberto Cristi
Co-Advisor

Dr. R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Weaknesses in the Fast Hessian detector utilized by the speeded-up robust features (SURF) algorithm are examined in this research. We evaluate the SURF algorithm to identify possible areas for improvement in the performance.

A proposed alternative to the SURF detector is proposed called rotated SURF (R-SURF). This method utilizes filters that are rotated 45 degrees counter-clockwise, and this modification is tested with standard detector testing methods against the regular SURF detector. Performance testing shows that the R-SURF outperforms the regular SURF detector when subject to image blurring, illumination changes and compression. Based on the testing results, the R-SURF detector outperforms regular SURF slightly when subjected to affine (viewpoint) changes. For image scale and rotation transformations, R-SURF outperforms for very small transformation values, but the regular SURF algorithm performs better for larger variations. The application of this research in the larger recognition process is also discussed.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

2D	two-dimensional
3D	three-dimensional
BOW	bag of words
BOVW	bag of visual words
BRIEF	binary robust independent elementary features
BRISK	binary robust invariant scalable keypoints
CIE	International Commission on Illumination
C-SURF	combined SURF
DOG	difference of Gaussian
FAST	features and accelerated segment test
FREAK	fast retina keypoint
GLOH	gradient location and orientation histogram
JPEG	Joint Photographic Experts Group
K-D	k-dimensional
LOG	Laplacian of Gaussian
NMS	non-maximum suppression
NN	nearest neighbor
NNDR	nearest neighbor distance ratio
ORB	oriented FAST and rotated BRIEF
PCA	principal component analysis
RANSAC	random sample consensus
RGB	red green blue color model
R-SURF	rotated SURF
U-SURF	upright SURF
SIFT	scale invariant feature transform
SURF	speeded-up robust features
XYZ	quasi-equal blue color model
YUV	one luminance two chrominance color model

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Automatic recognition of objects from images and video is a growing field designed to assist and ultimately replace tasks performed by humans (who require extensive training and familiarization) and provide the information in near-real time. For example, the United States Navy conducts counter-piracy and smuggling as well as an array of combat and non-combat operations that require the identification of vessels at sea. In order to make the recognition process automatic, a robust algorithm is required that is resistant to problems associated with computer vision, which includes object position, orientation, scale, intensity, surroundings, and partial occlusion.

There are four basic functions associated with feature-based algorithms such as the speeded-up robust features (SURF) or scale invariant feature transform (SIFT). They are (in order of occurrence) feature point detection, feature orientation assignment, feature descriptor generation, and feature matching [1], [2], [3]. SURF and SIFT are widely utilized as applications requiring feature extraction that are partially resistant to image rotation, affine (viewpoint) transformation, compression, illumination, and scale transformations [1], [2], [4]. Newer algorithms that use binary descriptors such as oriented fast and rotated brief (ORB), binary robust invariant scalable keypoints (BRISK), and fast retina keypoint (FREAK) provide slightly lower to comparable performance with significant computational speed improvement in both descriptor generation and matching [5], [6], [7].

Since the original goal of this work was to find the best detector for maritime vessel recognition, the research focus was on non-binary detection algorithms that prioritize performance over speed. In evaluating SURF versus SIFT, SURF was a better option for a number of reasons. SURF has been determined to have comparable performance to SIFT in regards to image transformations, with some advantages going to both algorithms depending on the transformation and test conditions [8], [9], [10]. Second, an open-source version of SURF was available [11] with a liberal license that allowed us to modify and test the algorithm [12]. Last, SURF is much faster than SIFT [8], [9], [10]. Although speed is not our main concern, since the two algorithms have

comparable performance efficiency in the implementation could ultimately play a role in the effectiveness of a maritime image recognition implementation. In order to evaluate the SURF algorithm, a basic description of how it works is necessary.

The SURF algorithm first converts an image to grayscale and computes an integral image [1], [2], [3], which is a cumulative summation of the rows and columns of the image matrix. From that point, the difference of Gaussians (DOG) scale-space representation is calculated by using larger versions of the Gaussian second derivative filter kernels [1], [2], [3]. A convolution filtering operation is then applied to the integral image utilizing a rapid method of four array accesses and calculations for each box filter [1], [2], [3]. The sampling location and different spatial sampling frequencies are used within each octave to form four DOG in each [1], [2], [3].

Using the second derivatives of Gaussians, Bay formed a Hessian matrix estimation (Fast Hessian) and assigned its determinant to the pixel at the filter center [1], [2], [3]. With the Fast Hessian determinants now represented in scale space, a three-dimensional search is performed in space and scale space within each octave to find interest points that are local maxima and also exceed a user-defined threshold [1], [2], [3]. Following interest point determination, Bay used a curve-fitting operation to provide additional sub-pixel interest point location accuracy [1], [2], [3]. For each interest point, an orientation vector is determined by taking the Haar wavelet responses in the x and y directions and determining the orientation with the highest concentration of Haar responses [1], [2], [3]. Lastly, Haar wavelet filter kernels are aligned with the orientation vector, and responses are determined within each sub-region [1], [2], [3]. The responses and their absolute values are summed within each sub-region to give four descriptor entries, and these are places in a vector for all 16 sub-regions to give a descriptor of length 64 [1], [2], [3].

We propose to use a new set of Fast Hessian detector filters obtained by rotating the original filter kernels 45 degrees counterclockwise. The main reason behind this choice is to introduce filters which orient the detector to an angle of rotational detection that is weak in the original SURF algorithm [8], [13]. A significant benefit of such a

choice of filters rotated by 45° is the ability to retain the speed advantage designed into the original SURF algorithm [3], [14]. This detector version was named rotated SURF (R-SURF). The rotated detector versus regular detector for the D_{xx} kernel having filter size 15 is shown in Figure 1.

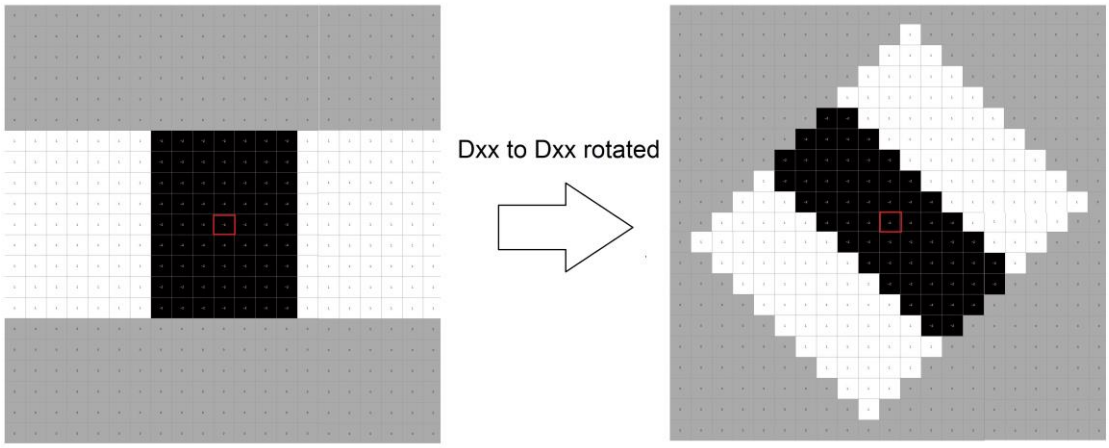

Figure 1. Rotated SURF box filter kernels for D_{xx} kernel filter size 15 (left side original SURF, right side rotated).

A standard testing method for detectors developed by Mikolajczyk et al. was used to evaluate the modified detector [15]. Eight image test sets with known homography and evaluation MATLAB code were obtained, which allows evaluation of detectors for images with transformations in affine (viewpoint), illumination, scale and rotation, JPEG compression, and blurring [15], [16], [17].

Results show the R-SURF outperforms the original SURF detector in its invariance when subjected to image blurring, illumination changes and JPEG compression. Results also show the R-SURF detector outperforms regular SURF slightly when subjected to affine (viewpoint) changes. For image scale and rotation transformations, R-SURF outperforms SURF for very small transformation values only, and the regular SURF algorithm performs better for larger values.

The method of rotating box filters in the Fast Hessian shows promise as a detection method. With testing showing performance increases, there may be additional

improvements that can be obtained by adjusting the filter kernel size or proportions. A detector that provides maximum invariance to changes that result from image transformations is desirable because the follow-on portions of the recognition process are affected by the detector quality [3]. An application of high interest to the DOD is automated maritime vessel recognition, where such work could contribute to increased reliability in the overall process

.

LIST OF REFERENCES

[1] H. Bay, Tuytelaars, T. Tuytelaars and L. V. Gool, "SURF: speeded-up robust features," *in European Conf. on Computer Vision*, Graz, Austria, 2006.

[2] H. Bay, A. Ess and L. Van Gool, "SURF: speeded-up robust features," *Computer Vision and Image Understanding (CVIU),* vol. 110, no. 3, pp. 346–359, 2008.

[3] H. Bay, "From wide-baseline point and line correspondence to 3D," Ph.D. dissertation,: Eidgenössische Technische Hochschule Zurich, Zurich, Switzerland, 2006.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," *in IEEE Int. Conf. on Computer Vision, Barcelona*, Spain, 2011.

[6] S. Leutenegger, M. Chli and R. Siegward, "BRISK: binary robust invariant scalable keypoints," *in IEEE Proc. Int. Conf. in Computer Vision, Barcelona*, Spain, 2011.

[7] A. Alahi, R. Ortiz and P. Vandergheynst, "FREAK: fast retina keypoint," *in IEEE Conf. on Computer Vision and Pattern Recognition), Providence*, RI, 2012.

[8] L. Juan and O. Gwon, "A comparison of SIFT, PCA-SIFT and SURF," *Int. Journal of Image Processing)*, vol. 3, no. 4, pp. 143–152, 2009.

[9] K, Y. N. Khan, B. McCane and G. Wyyill, "SIFT and SURF performance evaluation against various image deformations on benchmark dataset," *in Int. Conf. on Digital Image Computing Techniques and Applications*, Noosa, QLD, Australia, 2011.

[10] J. Bauer, N. Sunderhauf and P. Protzel, "Comparing several implementations of two recently published feature detectors," *in Proc. of the Int. Conf. on Intelligent and Autonomous Systems*, Toulouse, France, 2007.

[11] D. Jan Kroon, "OpenSURF (including image warp)," 06 Oct.2010. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/28300-opensurf--including-image-warp-

[12] Open Source Initiative, "The BSD 3-clause license," [Online]. Available: http://opensource.org/licenses/BSD-3-Clause

[13] K, Y. N. Khan, B. McCane and G. Wyyill, "SIFT and SURF performance evaluation against various image deformations on benchmark dataset*," in Proceeding of the Int. Conf. on Digital Image Computing Techniques and Applications*, Noosa, QLD, Australia, 2011.

[14] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," *IEEE Proc. from the Int. Conf. on Image Processing*, pp. 900–903, 2002.

[15] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir and L. Van Gool, "A comparison of affine region detectors," *Int. Journal of Computer Vision*, vol. 65, no. 1/2, pp. 43-72, 2005.

[16] Visual Geometry Group, Katholieke Universiteit Leuven, Inria Rhone-Alpes, Center for Machine Perception, "Comparison of region descriptor," 16 Mar. 2006. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/research/affine/desc_evaluation.html#code

[17] University of Oxford, "Visual Geometry Group," 2004. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/data-aff.html

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my girlfriend, Sade, and my mom, Michelle, for their support throughout this process. They are always supportive and wonderful through challenging times. I would also like to thank my thesis advisors, Dr. Fargues and Dr. Cristi, for their great advice and hard work. Without their technical knowledge, attention to detail, and tough questions, this work would have been very difficult to complete. Last, I would like to thank the people who performed the preceding analysis which this thesis was based upon, especially those who made their work available in an open-source platform.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM DESCRIPTION

The United States Navy and Department of Defense are always looking for ways to identify either the enemy or items of interest. This process is crucial to almost all operations. Unit and asset geographical location restrict the reach of sensors and weapons and have traditionally required humans to conduct the identification and classification of items of interest. This problem gets much bigger when talking about oceans and waterways. The Earth's surface is 71% covered in water [1], and there are only a limited number of ships, aircraft, and submarines to provide sensor reach in such a vast region.

With the ever more expanding of use of unmanned vehicles, cheap sensors, and interconnected networks, it is necessary to design algorithms that automatically perform operations that used to be performed by humans. Aside from freeing up human resources for more complex tasks and enhancing their productivity, the breadth of information being processed currently (and what will be in the future) make it impractical to perform these duties manually.

This need for automation is particularly true for the U.S. Navy. The Navy's ability to automatically identify and classify maritime vessels from images in harbors and at-sea is a major force multiplier and capability enhancement tool for any mission. The Navy is already fielding MQ-8C Firescout autonomous aircraft that are designed to carry reconnaissance and surveillance sensors with data links to other units [2]. It stands to reason that existing sensors or future modifications will provide images that will need to have algorithms that detect and classify objects of interest. Additional unmanned vehicles deployed by the Navy include the RQ-2A Pioneer UAV [3], sub-surface MINI-ROVs [4], and sub-surface MAGNUM-ROVs which could all be utilized for gathering of sensor image data from their respective domains.

The problem with using images for this task is a need for the ability to discriminate using limited information. For example, if a $256 \times 256 \times 3$ color scheme with an uncompressed image is used, each visual pixel has a possibility of 256^3 combinations

[5]. There are essentially 256^5 (or over a trillion) different combinations that make up an image of fairly small size. Not all images come from color sensors, and a lot of algorithms utilize only greyscale image representations due to colors having high correlation with the intensity [6].

It is clear that an algorithm is needed to identify intensity patterns through common image transformations and also to provide a representation of an object so that it is useable for a follow-on matching algorithm. The speeded-up robust features (SURF) algorithm is an efficient and effective tool for describing interest points [7], [8], [9], [10]. There are many algorithm choices available, but the combination of robustness and efficiency provided by SURF make it a desirable algorithm to use as a baseline. Since SURF has been shown to not always perform as well as an earlier feature algorithm called scale invariant feature transform (SIFT) in its invariance to certain image transformations [8], [9], [10], there exists an opportunity to evaluate the algorithm and possibly propose changes that may result in a more effective feature detector and descriptor.

B. IDENTIFICATION OF NEED FOR RESEARCH

Image recognition and computer vision are important topics with a wide range of applications ranging from medical industry to security. In order for this technology to develop, multiple issues have to be addressed. The problem with video and camera images data is the bandwidth required to carry the data. There are multiple questions that arise when assessing the image recognition problem, including whether all frames or images need to be transmitted and whether the recognition algorithm needs to be applied to each frame.

Rainey et al. from the Space and Warfare Center (SPAWAR) Systems Pacific produced a brief for the 2012 National Geospatial Agency symposiums and workshops which presented an evaluation of feature-based detectors of satellite-based imagery of maritime vessels [11]. The authors employed the SIFT together with a bag of words (BOW) algorithm for images where the maritime vessel orientations between the two images are geometrically aligned [11]. A main take-away point from the authors was the

necessity for more research on feature extraction methods [11]. This conclusion seems especially true if the goal is to deliver useful data to algorithms like BOW or another learning algorithm that is used to classify images or objects based on feature descriptors and accompanying information.

Feineigle et al. analyze the process of optical ship recognition for harbor surveillance [12]. The focus of this work was in using the SIFT detector-descriptor combination to evaluate the feasibility of harbor surveillance with a geometric verification process to filter out false matches. The authors' geometric process involves taking each pair of matched features change in position, orientation, and scale and applying an iterative least-square fitting procedure to determine if the interest point affine translation was consistent with the database values [12]. The authors concluded that there was promise in using SIFT features for harbor surveillance, but results showed issues with lighting differences and scale changes [12]. This work, especially the geometric test used to determine false matches, was a very good start for developing automated maritime recognition methods. The progress of computer vision, with many new methods for detection, description, and matching, allows us to build on this work and develop methods that are best suited to a maritime environment.

Given the importance of a system that can identify and classify objects of interest from an image, there is a great need to continue to perfect current processes and devise new ones. A large portion of advances in computer vision and image processing work currently underway is occurring for civilian applications. There is a large need to translate this work into a form that is of interest to the United States Navy (and other services) so that algorithms can be adapted or developed to fit military-based needs of future forces. This goal of this research is to improve the descriptor in the SURF algorithm, which is a small part of the larger system of image recognition.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

The goal of this section is to present an introduction to SURF, a widely-used feature detection and descriptor algorithm that is the focus of this research. This algorithm, introduced by Bay et al. in [6], is well known not only for its effectiveness in identifying image features but also for efficiency in practical applications [8], [9], [10]. In the remainder of this thesis, SURF is the algorithm modified and tested, but feature-based algorithms that preceded SURF as well as those that developed after are discussed to assist in gauging the applicability to maritime recognition. Common methods for SURF and other algorithm feature matching are covered to provide an end-state for the algorithm modifications. The consolidated descriptions in this chapter are also intended to assist in future work related to this topic.

The goal of a feature extraction algorithm is to deliver descriptors of features to a follow-on matching algorithm. The desire is for these features to be robust to change in the image characteristics so that they are repeatable and useful in matching, with the goal to identify specific objects present in an image.

A. REVIEW OF PAST RESEARCH

Feature detection and matching approaches are fundamental in many different computer vision applications, such as image stitching and three-dimensional (3D) model reconstructions [13]. Although some algorithms have properties making them better suited for specific uses, it is necessary to have descriptors that are distinctive in relation to different features as outlined in [13], [14]. Additionally for detectors, a good *repeatability* of corresponding regions in the presence of variations in scale, viewpoint, illumination changes, compression, rotation, blurring, partial occlusion, and background clutter avoidance is desired as described in [13], [15].

1. Scale-Invariant Feature Transform

The basic SIFT algorithm has four stages: scale-space extrema detection, keypoint localization, orientation assignment, and keypoint descriptor generation [16]. These are

the same general functions that the SURF algorithm performs, with the exception being the details within each portion [6], [17], [18].

Scale space, which is a technique designed to smooth images with repeated Gaussians to progressively remove smaller spatial frequency details, is used in order to establish a measure of scale invariance [19]. If there is an object, or feature, that is captured by a picture (say) four feet in distance away, the goal is to match the same feature with a picture taken 15 feet away. By intuition, one would expect that a closer image object would occupy more pixels than the object contained within an image obtained from a higher distance. Thus, a need to find a procedure which allows identical identification of the same objects at different scales is a motivating factor for the use of scale space.

To achieve this goal, the SIFT algorithm uses a method drawn from Lindeberg's scale-space theory work [16], [19]. The problem faced is that there is only access to an intensity image represented by a given scale for an individual image not known a priori. As a consequence, to make the extracted features scale invariant, a scheme is needed that effectively "searches" the image at varying scaling levels. The result is the representation of an object at different scale levels by removing spatial frequency components, "removing" smaller irrelevant details. This interest point is then used for follow-on orientation and descriptor generation [19].

This multiscale representation is obtained by convolving the image with a Gaussian kernel [16]. The main motivation for choosing a Gaussian kernel for convolution is that different scale representations are obtained just by changing the standard deviation parameter. Another reason behind choosing a Gaussian kernel is that it only removes unwanted image details without introducing artifacts [16], [19]. A Gaussian pyramid is built with the highest resolution at the base and lowest resolution at the top [16].

This pyramid scale space representation of an image is further enhanced by Lowe's use of a difference of Gaussian (DOG) kernel to generate the scale space [16], which preceded Bay's SURF algorithm [6], [16]. In Lowe's scheme, different resolution

pyramid levels are grouped into octaves, as shown in in Figure 1 on the left [16]. To produce the final octave construct, adjacent scales are then subtracted to form the DOG assembly of scale space as shown in Figure 1 on the right [16].

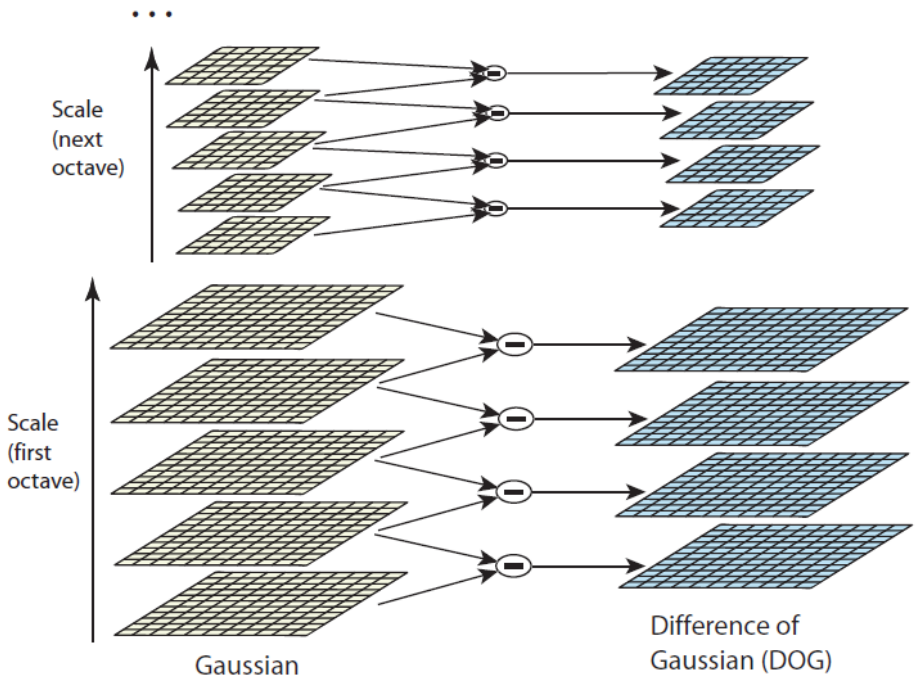

Figure 1. Difference of Gaussian (DOG) for SIFT algorithm, from [16].

For follow-on octaves, the image is down-sampled by a factor of two, and the Gaussian filtering and DOG process is repeated [16]. Additionally, Lowe doubles the base octave size in the scale-space pyramid through linear interpolation in order to retain higher spatial frequencies after smoothing [16]. At the end of this process, multiple octaves are obtained, each containing DOG spatial intensity image two-dimensional (2D) representations. The intent of the DOG is to approximate the scale-normalized Laplacian of Gaussian (LOG), which can then be used to detect interest points in scale-space [16].

At this point, the 3D search for extrema points is conducted using the 26 pixel neighbors on a $3\times3\times3$ cube of scale space [16]. After the identification of extrema, sub-pixel accuracy is determined using quadratic interpolation [16]. Interest points resulting from the DOG strong edge responses are removed according to

$$\frac{Tr(H)^2}{\det(H)} < \frac{(r+1)^2}{r}, \tag{1}$$

where H is the Hessian matrix and the variable r represents the desired ratio of the larger to the smaller eigenvalue [16]. The point of the thresholding operation is to ensure that the principal curvature is below a predetermined value, which allows discrimination between interest points that represent features. The result of thresholding is a removal of incorrect anomalies that occur at the DOG representation edges [16].

Next, a feature orientation assignment is performed after the selection of keypoints of interest. Based on the keypoint strength or scale, the Gaussian smoothed image with the closest scale is selected. Lowe uses the pixel immediately below and to the right of a sample pixel in equations that determine the gradient orientation and magnitude for that particular sample [16], [20]. These sample pixels are from locations in the region surrounding an interest point. From the calculated sample pixel gradient orientations, a 36 bin histogram is formed. The histogram maximum and any values within 80% of its maximum are used to create keypoints with corresponding orientations [16], [20]. If there is one detected interest point with a peak and two gradient bins that are within of 80% of peak magnitude, then three keypoints are created at the same location and different orientations [16]. Additionally, a parabola is fit to the peaks in order to interpolate the actual orientation for higher accuracy [16].

The last part in the SIFT algorithm is the generation of a local descriptor [16]. A region surrounding the interest point is constructed and divided into sub-regions [16]. A Gaussian blur is applied to the region based on the keypoint scale. The gradient magnitudes and orientations are obtained for all sub-regions, and a Gaussian weighting function is applied to avoid rapid changes between regions due to small positional shifts [16].

Each of the sample point gradients and the interest point location are rotated and aligned to the reference axis determined in the previous orientation step to achieve a level of rotation invariance for the feature [16]. The sample point gradient magnitudes and orientations are summed into orientation histograms to represent a sub-region.

Interpolation by distributing values from adjacent bins into neighbors is applied to the histogram bins to avoid steep sample boundary effects [16]. The region surrounding the interest point is divided into 4×4 sub-regions, or 16 total, each with eight histogram bins. This division and histogram representation results in a feature descriptor array of size 128 [16]. An example of this descriptor generation process for SIFT is shown in Figure 2. This example utilizes only a region of interest with 2×2 sub-regions (as opposed to 4×4 used in the SIFT algorithm) for ease of viewing [16].

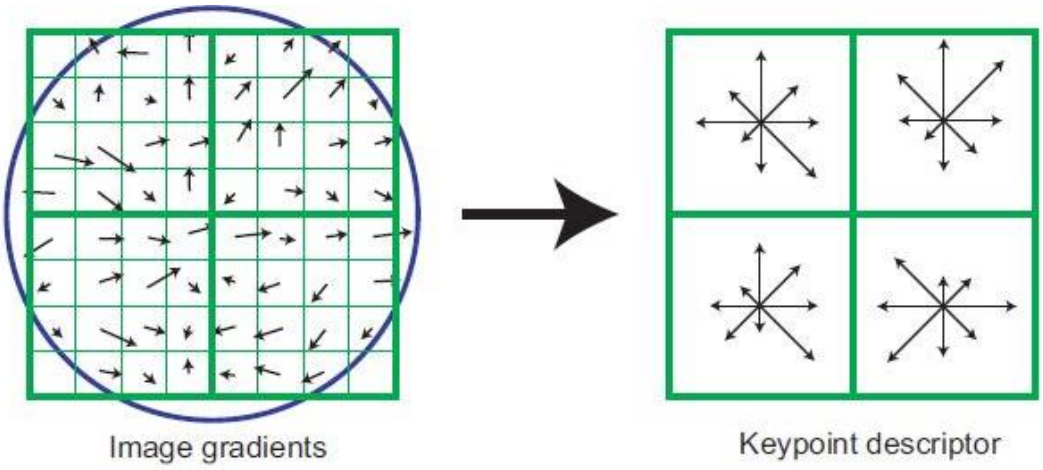

Figure 2. Generic SIFT keypoint descriptor generation for 2×2 sub-regions from 8×8 samples, from [16].

A proposed modification to the SIFT descriptor was proposed by Mikolajczyk and Schmid called the gradient location and orientation histogram (GLOH) [14]. In this approach, instead of using 4×4 sub-regions, the authors divide the descriptor generation area surrounding the interest point into three bins in the radial direction and eight different angular bins (except the central bin) for a total of 17 bins [14]. There are also 16 different gradient bins within each location bin, so the total histogram for the descriptor has 272 bins [14]. Further, Mikolajczyk and Schmid reduced the descriptor size through PCA and testing, with the 128 highest value eigenvectors corresponding to bins being retained [14]. Finally, the authors use a Canny edge detector to find the position and orientation of edges in the region and quantize the descriptor down to nine bins as shown in Figure 3 [14]. The descriptor region shown has radius values equal six, 11, and 15 (on

a log-polar grid) in the figure for bin locations, each containing four orientation bins at angles 0°, 45°, 90°, and 135° [14]. This process results in a total descriptor of size 36 (nine locations which each have four orientation bins) [14].

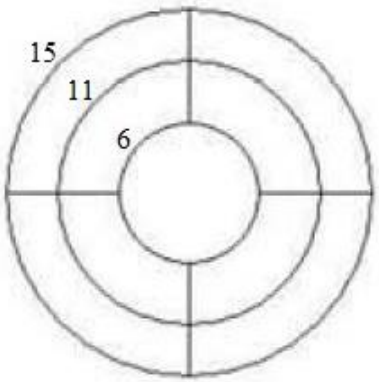

Figure 3. Orientation of GLOH descriptor location bins, from [14].

2. Speeded-up Robust Features

A SURF algorithm overview, which is the principal focus of this study, with particular emphasis on the concepts of interest point detection and description is presented in this section. In SURF, most fundamental concepts are along the same lines as for SIFT [6], [16]. The main difference is a comparatively efficient implementation that makes the algorithm more suitable for near-real time applications or processing larger numbers of images.

a. Interest Point Detection

The first step in image classification is the detection and characterization of "points of interest" later used to characterize feature information [6]. In SURF, this requires a series of steps to be covered below.

(1) Detection step one: conversion to grayscale. The SURF algorithm works with a 2-D image grayscale matrix; thus, any color image must be converted into the proper intensity format [6].

(2) Detection step two: integral image computation. The integral image's purpose is to facilitate rapid box filtering, which was first introduced in computer vision

by Viola and Jones in 2001 [21]. In images, the integral image step corresponds to a cumulative pixel intensity sum from locations left and above a target pixel in the image. The integral image at indices (x, y) is defined as

$$ii(x, y) = \sum_{x' \le x, y' \le x} i(x', y'), \tag{2}$$

where $i(x', y')$ represents the intensity of pixels located at all indices satisfying the inequalities $x' < x$ and $y' < y$ [21]. As shown in Figure 4, the integral image can be computed on the entire image very rapidly by cumulatively summing first all rows of an intensity image, then all columns.

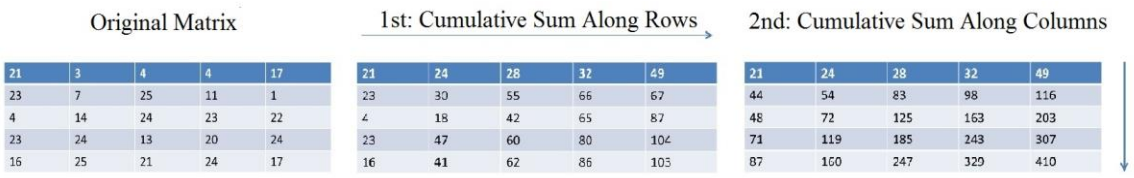

Figure 4. Example of integral image summation.

(3) Detection step three: box filtering. This method is an efficient process to calculate the Hessian matrix at a given pixel (and the follow-on Hessian determinant) as is described in the next step. The efficient box filtering process is also used to apply Haar wavelet kernels for the orientation and descriptor generation steps described later [6], [17], [18]. Box filters use a box (or boxes) defined relative to a given center pixel, with each box having given weights [18], [21], [22]. For this class of filters, the convolution operation linearly combines the pixel intensity sum for that box [23]. In each situation, the kernel is developed using boxes with each pixel in the box having equal weight in order to conduct the filtering on an image, with the center being the pixel of interest [23]. A basic example illustrating the use of a convolution kernel to filter an image is highlighted in Figure 5. This is not the box filter utilized in the SURF algorithm but rather is a basic example to visually show how a generic convolution kernel works [17].

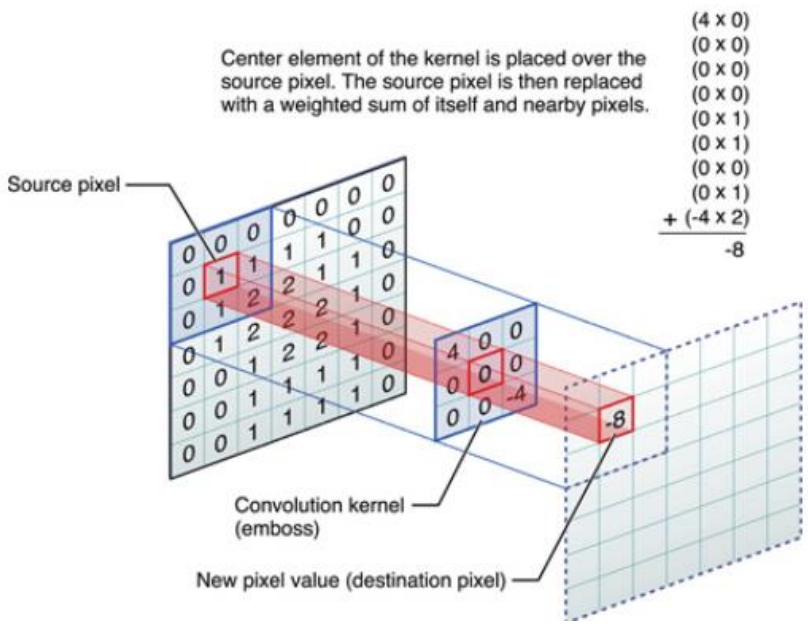

Figure 5. Generic example of convolution filtering [23].

The problem with using a convolution kernel for filtering is that it is not computationally efficient. A 9×9 convolution kernel filter requires 81 array accesses and operations for every center pixel [18]. The regular convolution method computational load increases along with increasing filter sizes (increasing filters are necessary in the scale-space analysis which is discussed in Section II.A.2.a(4)) [24]. Since a more efficient method is desired, the integral image properties are used for a more efficient "box filter" kernel output calculation than provided by regular convolution with the intensity image. A visual reference describing a rapid method for calculating box integrals on an integral image [21] is provided in Figure 6.

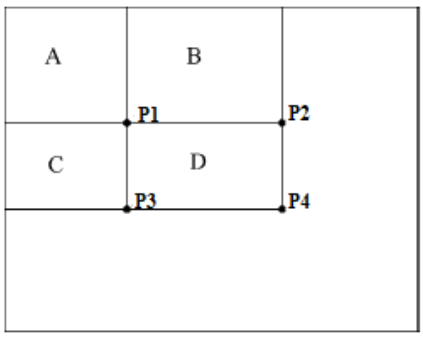

Figure 6. Pixel accesses for rapid box filtering using integral image, from [21].

Refer to Figure 6 and assume that areas labelled A, B, C, and D are adjacent non-overlapping rectangular image sections in the integral image defined by Equation (2) [21]. The box labelled A of pixels includes the edge pixels on the top left. The variables P_1, P_2, P_3, and P_4 represent the integral image pixel values at bottom right corner pixels of box A, B, C, and D, respectively [21]. The sum of all intensity pixel values contained in *original intensity image* box A can be obtained by the value at *integral image* pixel P_1. Similarly, the sum of all pixel values contained in box B is obtained by taking the difference $P_2 - P_1$. For box C, we use $P_3 - P_1$. Finally, for box D the formula is $P_4 - (P_2 - P_1) - P_3$ [21]. This method allows for an efficient convolution process between the image and the box kernel [18], [21]. This result also shows that only four pixel accesses and operations are needed to calculate the convolution operation for a single box regardless of its size [21]. This approach makes the convolution box filtering operation invariant with respect to the filter kernel size and drastically reduces the computational complexity (the larger the filter kernel, the more the proportional decrease from baseline box filtering). As will be seen, this can easily be extended to box filtering kernels used in the SURF algorithm; multiple boxes with different weights can be used to represent a kernel. The computation requires only four array accesses and operations per box [6], [17], [18]. In the case where the box filter edge falls outside the integral image, values assigned to the points left and above the image are set to zero. The points to the right and below the images are set to the integral image closest edge pixel intensity [25].

An operation of interest in the SURF algorithm is the Hessian computation to determine points of interest. In particular, *Fast Hessian* is a term that was used by Bay when he first proposed the SURF algorithm to describe his method for determining the Hessian determinant of a center pixel using an integral image operation combined with the rapid box filtering convolution approach illustrated in Figure 6 [6], [17]. First, we describe how the Hessian is determined for a pixel and its surroundings and follow with the "Fast Hessian" approximation explanation.

The Hessian matrix $H(f)$ is defined as the square matrix of second-order partial derivatives of a function $f(x, y)$, as proposed by Hesse [26]. Since the SURF algorithm

operates in 2D space (based on the 2D intensity image), the Hessian matrix is given by [26]

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}. \quad (3)$$

The Hessian matrix determinant is designed to measure pixel intensity level curvature in the region around a given point, which is used for detection of "blob-like" areas in the SURF approach [18]. In the particular SURF algorithm application, the function f is the result of the image intensity level convolution with an approximated Gaussian [6], [17], [18]. Each second-derivative entry in the Hessian are constructed as a box filter that is applied directly to integral image sample points using the rapid method in Figure 6. The result of this filtering technique is to calculate surrounding responses about a given pixel for the original intensity image [6], [17], [18]. Ideally, the Hessian determinant value is larger for a pixel surrounded by regions with high Gaussian curvature, which is the product of two principal curvatures (i.e., 2D curvature) [15], [27]. The Hessian matrix determinant approximation (or Fast Hessian as termed by Bay), is used to determine the feature center location or interest point [6], [17], [18]. Bay describes this process as detecting “blob like structures” [6], [17], [18]. Additionally, the Laplacian $L(f) = \partial^2 f / \partial x^2 - \partial^2 f / \partial y^2$ is calculated for faster matching but is not used in the descriptor generation or interest point discovery [6], [17], [18].

We return to more specifics about scale-space for SURF in Section II.A.2.a(4). Now that it is established that the SURF algorithm uses an approximate Hessian matrix determinant of the second-derivative Gaussian-filtered image to determine an interest point, we introduce how the “Fast Hessian” works. Bay proposed using the previously described box filters as kernels for determining the second-order Gaussian partial derivatives included in the Hessian matrix [6], [17], [18]. Multiple 9×9 filter kernel examples are shown in Figure 7 with the actual discretized and cropped Gaussians on the

left and the box filter equivalent approximations used in SURF on the right [6], [17], [18].

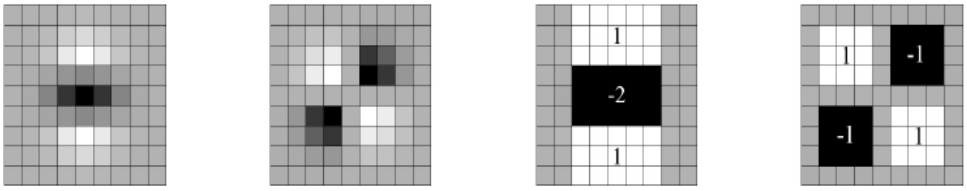

Figure 7. Example of Gaussian kernels (left) vs. box integral kernel approximations (right), from [18].

(4) Detection step three: scale space generation. As discussed in SIFT s*cale space* generation in Section *II.A.1*, the structure of this multi-spatial representation is used in order to establish a measure of scale invariance in the SURF application [6], [17], [18]. We now do a brief comparison between the SURF and SIFT algorithm methods for generating scale space to highlight the differences in efficiency utilized by Bay for SURF [18].

In the SURF algorithm, the approach taken by Bay is slightly different from the one taken by Lowe. We do a comparison in the next two paragraphs to highlight the main differences, and then return to the SURF algorithm description. As shown on the left in Figure 8, Lowe reduced the image size for each octave before the Gaussian filter kernel is iteratively applied to scales within each octave to obtain the Gaussian smoothed intensity image representation [16]. Next, the DOG representation is obtained by taking the difference between two adjacent Gaussian filtered images within the octave to form the DOG representation [16]. In contrast, the SURF algorithm only up-scales the filter and applies each filter to the exact same integral image once (which is an approximation of constructing the DOG with lower octaves down-sampled after filtering) [6], [17], [18]. A good visual contrast between both scale space generation processes followed by the SIFT (left) and SURF (right) algorithms is highlighted in Figure 8.

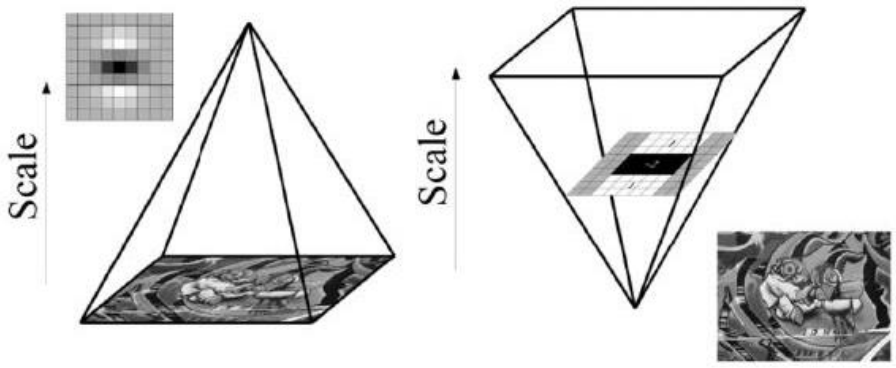

Figure 8. Comparison of SIFT (left) versus SURF (right) scale space generation, from [24].

For the SURF algorithm, the box filter property combined with an integral image allows each box of a given weight in the filter kernel to be calculated with only four pixel accesses and operations for each sample pixel as discussed in Section II.A.2.a(3). This step greatly reduces the computational cost associated with generating the image scale-space representation [6], [17], [18], [24]. To summarize differences between the two algorithms, SIFT requires an intensity image first to be up-sampled by two, then the image has a Gaussian filter applied iteratively five times. After each Gaussian filer application, a resulting image is retained [16]. Next, the differences between adjacent Gaussian filtered images are computed to create four DOG approximations within a single octave as illustrated in Figure 1 [16]. This step is then repeated for higher scale octaves by down-sampling the intensity image and again applying Gaussians to the scales within the octave [16]. In contrast, SURF uses the integral image of the intensity image and applies four increasing size box filtering operations with the rapid calculation method shown in Figure 6. No subtraction of filtered intensity images is needed [6], [17], [18]. For the higher octaves, two previous filter sizes are used (of four), so only two additional image filtering operations are needed to form an octave. Each successive scale level has the Fast Hessian box filter kernel applied at half the sampling (both vertically and horizontally) [6], [16], [17], [18]. A large portion of the computational cost savings in SURF over SIFT is accounted for in the differences in how the scale space DOG approximations to the LOG are generated [16], [18].

Continuing with SURF, and referring to Figure 7, we see that the Gaussian kernel box filter approximation is not an exact representation of a Gaussian. In order to compensate for this difference, Bay proposed a formula for calculating the Fast Hessian matrix determinant that weights the diagonal second derivative with a factor of 0.9

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2, \tag{4}$$

where D_{xx} is the horizontal second derivative box filter response for a given integral image center pixel, D_{yy} is the vertical filter response, and D_{xy} is the diagonal filter response [6], [17], [18].

The equivalent scale for each second-order derivative Gaussian kernel corresponds to the base scale or smallest Gaussian kernel by the approximation

$$\sigma_{approx} = CurrentFilterSize \times \frac{\sigma_{BaseFilter}}{BaseFilterSize}, \tag{5}$$

where σ is the Gaussian standard deviation represented in the second-order Gaussian derivative approximation box filters in the SURF algorithm [18].

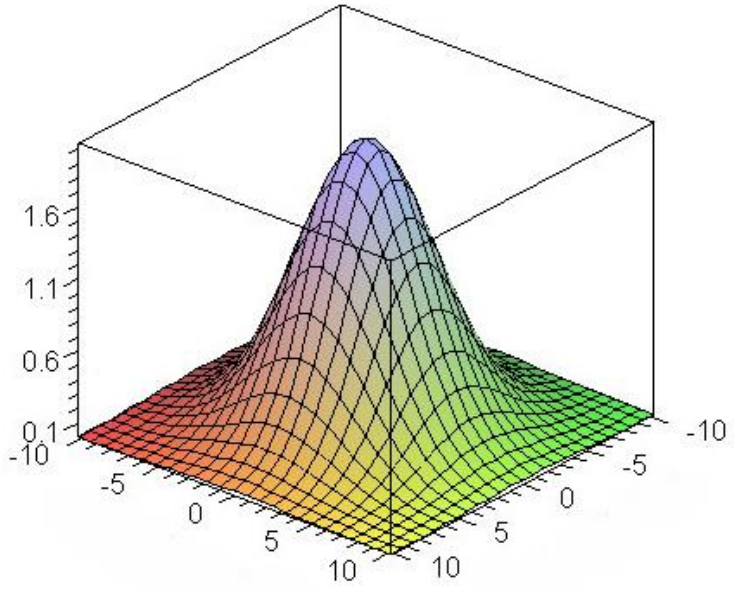

Figure 9. Generic two-dimensional Gaussian example, from [28].

The base 9×9 SURF algorithm filter corresponds to a Gaussian scale (standard deviation) of 1.2 [6], [17], [18]; thus, the equivalent Gaussian scale for any filter can be determined by using Equation (5). Refer to Figure 9 for a visual representation of a generic two-dimensional Gaussian as a reference. This is not the exact Gaussian used in SURF; rather the SURF algorithm contains three filter kernels that are Gaussian second-order derivatives approximations [6], [17], [18]. As box filter scale or standard deviation is increased in the SURF algorithm, the Gaussian function estimation becomes larger in order to compute the image at lower resolution [18].

In order to maintain a consistent ratio of box filter Gaussian kernel height and width as the filter increases in size, care must be taken to ensure everything is proportionally increased given that the SURF algorithm is operating on discrete pixels [18]. The reason behind desiring filters that maintain equivalent size ratios is to enable generation of a scale space so that each level corresponds to an equivalent Gaussian approximation [18]. The increase in filter sizes is shown in Figure 10 and represents a Gaussian scale approximation of 1.2 (left filter size nice) and a scale of two (right filter size 15) for the second-order Gaussian derivative filter in the vertical (y) direction [18].

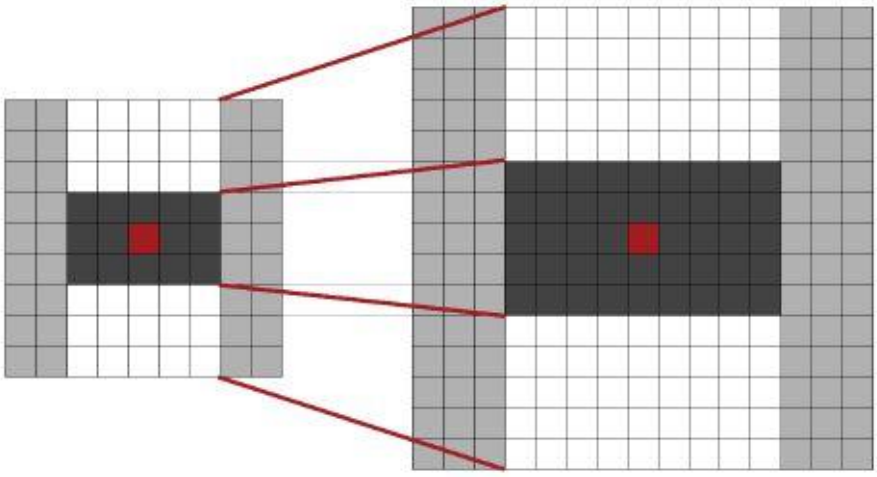

Figure 10. Growth of a box filter for D_{yy} Gaussian kernel approximation, from [18].

Next, the algorithm creates octave DOG approximations with the results from filtering the image with increasing size box filters. These can be represented as 3D pyramids where a 2D image is represented at different Gaussian scales as in Figure 8. Each octave is constructed with a 2D integral image blurred by filters of increasing size to create each of the four scale levels internal to the octave, with the third dimension variable considered the second-order derivative Gaussian standard deviation (scale) used to blur the image [6], [17], [18]. Each octave is sampled at half the rate of the previous, resulting in a smaller number of samples for higher octaves [6], [17], [18]. For example, a downsampling by two of an image representation being sampled at every pixel means that every other column is sampled horizontally and every other row is sampled vertically. The filter sizes used to construct the DOG representation for the octaves are highlighted in Table 1 (note that the initial sampling rate is user-selected) [25].

Table 1. Box filter kernel size used to generate octave DOG representation in SURF, after [6].

	Gaussian Approximation Box Filter Size				
Octave 1	**9**	**15**	**21**	**27**	Sampled @ Initial Sampling Rate
Octave 2	**15**	**27**	**39**	**51**	Sampled @ 2× Initial Sampling Rate
Octave 3	**27**	**51**	**75**	**99**	Sampled @ 4× Initial Sampling Rate
Octave 4	**51**	**99**	**147**	**195**	Sampled @ 8× Initial Sampling Rate
Octave 5	**99**	**195**	**291**	**387**	Sampled @ 16× Initial Sampling Rate

Now that octaves are generated and DOG approximations within each octave have been created by the SURF algorithm, a 3D search for the local Hessian determinant maxima for “interest points” location can be performed [6], [17], [18].

(5) Detection step five: 3D search for interest points. This search is conducted using a non-maximum suppression (NMS) approach over the neighbors from 2D space on the two middle scales of each octave and also the adjacent scale-space neighborhood [18]. When focusing the search around a single pixel, a $3\times3\times3$ scale space neighborhood is used to determine if the sample is a local maximum within that search region [6], [17], [18], [24]. A pictorial representation of the adjacent pixels in space and scale space is provided in Figure 11. The center pixel is considered a local maximum if it has the highest intensity in the search region [18]. The center pixel is then compared to a user-defined threshold, and if it exceeds the threshold and is a local maximum, then the pixel is considered to be an interest point [18].

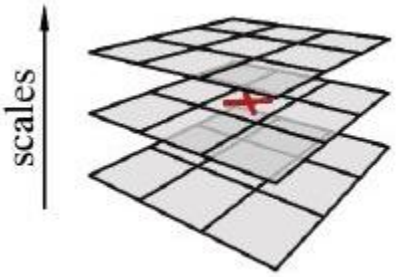

Figure 11. Example of 3D scale-space search around pixel of interest, from [24].

Next, we introduce NMS, which is applied as part of the local maximum search process [18]. In the SURF algorithm the local maximum is found by "searching" a 3D region (again x, y, and $scale$) surrounding the center pixel with dimensions

$$(2N+1)\times(2N+1)\times(2N+1), \tag{6}$$

where N is the number of pixel directly in any axis direction ($\pm x$, y, or σ) contained in the search box, where σ is the scale axis variable [18]. As shown in Equation (6) and Figure 11, the smallest possible neighborhood of nearest neighbors to the pixel of interest is with $N=1$, which yields a neighborhood of size $3\times3\times3$ as used in the SURF algorithm [18]. A straightforward implementation of this algorithm in computationally expensive since the search requires more calculations than desired or necessary, so SURF

uses an efficient algorithm to search through scale-space instead. This fast method is highlighted by Bay [18] and Neubeck and Van Gool in [29]. This more-efficient version of NMS is referred to as a "block algorithm." This algorithm divides the search area into blocks with sizes $(2N+1)\times(2N+1)\times(2N+1)$ [18]. These non-overlapping blocks are searched to find a single local maximum. If a local maximum in the block is found, then all other pixels can be ignored in future searches because that pixel already has been determined to have the largest value in its neighborhood [29]. The complexity savings from using this technique are displayed in Table 2.

Table 2. 3D search method efficiency comparison, from [18].

Algorithm	worst case	average case
1D Straightfwd	N	$\approx 1+\ln(N)+\ln(2)$
1D Block	$2-\frac{1}{N+1}$	$\approx 1+\ln(2-\frac{1}{N+1})$
2D Straightfwd	N^2	$\approx 1+2\ln(N)+\ln(2)$
2D Block	$4-\frac{4}{N+1}$	$\approx 1+2\ln(2-\frac{1}{N+1})$
3D Straightfwd	N^3	$\approx 1+3\ln(N)+\ln(2)$
3D Block	$\approx 8-\frac{12}{N+1}-\frac{6}{(N+1)^2}$	$\approx 1+3\ln(2-\frac{1}{N+1})$

The NMS approach result is that all local maxima retain their values, all other points are set to zero, and any local maximum not exceeding the user-defined threshold value is also set to zero [18]. In the SURF algorithm, there are four DOG representations per octave from the Fast Hessian filtered results as described in the previous section [18]. The top and bottom (highest and lowest scale) DOG within an octave are only used for reference in the interest point search for the two middle scales [18]; thus, only the second and third layers in each octave can determine a point of interest, but a higher Hessian determinant value in the first and fourth layers can eliminate an interest point from being identified as local maximum [18].

(6) Detection step six: sub-pixel accuracy determination. The attainment of sub-pixel accuracy takes place via a quadratic interpolation method [6], [17], [18]. The interest point direct neighbors are considered (in both the image and scale space) and an

interpolated position is found [6], [17], [18]. From the previous search algorithm, it is already known that the 3D neighborhood of Hessian determinant values are lower than the value obtained for the interest point, so the conclusion is that the interpolated value falls within a 0.5 pixel range of the original interest point location and not closer to an adjacent pixel [18]. A one-dimensional parabolic fitting function used between adjacent pixels is seen in Figure 12.

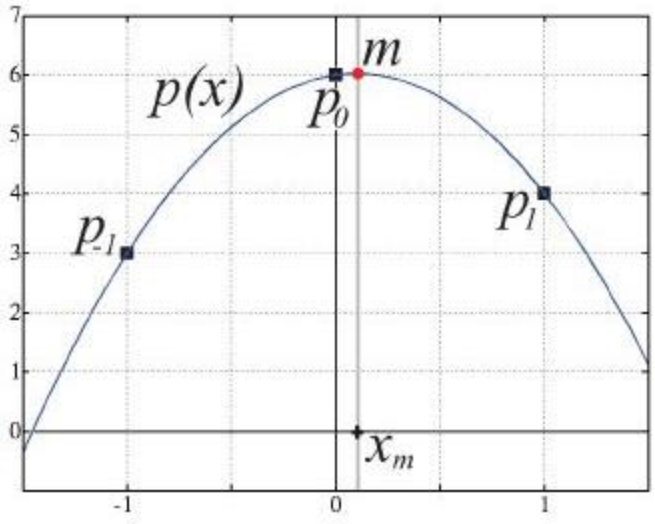

Figure 12. Example of one dimensional sub-pixel interpolation of an interest point, from [18].

In the example case in Figure 12, the original local maximum was found at pixel p_0, and the neighboring pixels p_{-1} and p_1 along the axis are used for quadratic fitting. The interpolated maximum can be offset from the original interest point pixel location. The maximum is determined to be at point m on the parabola, which is located at a fractional distance between p_0 and p_1 [18]. A new sub-pixel position is assigned to the detected interest point based on the interpolated location m. The same method is used for the $y-$direction to achieve a sub-pixel position for the interest point [18].

At the conclusion of the interpolation steps described above, each interest point with its corresponding strength (represented by the Fast Hessian determinant value) and sub-pixel location have been determined.

b. Interest Point Description

For each point of interest identified in the previous subsection, we need to define set of descriptors which characterizes the area surrounding each point. The process used in the SURF algorithm is described in the steps below.

(1) *Description step one: orientation assignment.* The orientation step goal is to provide a directional value for each feature that can be used to assist in calculating a partially rotationally-invariant descriptor. The orientation value is also an algorithm output for optional matching scheme use [18], [25]. In the regular SURF algorithm, feature orientation is calculated for the area surrounding the interest point in order to achieve some degree of detected feature rotation invariance. In order to attain an orientation value, a region of radius $6s$ (with s representing the Fast Hessian determinant value at a given interest point) around the interest point location is defined [6], [17], [18]. Haar wavelets are generated as shown in Figure 13. During filtering, these kernels are applied with the same fast box filter convolution process shown previously in Figure 6 [6], [17], [18], [25].

Next, the SURF algorithm uses the Haar filter kernels to determine responses in the (x, y) plane [6], [17], [18]. Furthermore, the search region size is dependent on the variable s previously defined, and the Haar filter kernel size is also dependent on s with each square kernel side having length equal to $4s$ [18]. Additionally, specific pixel responses used for orientation determination are sampled at vertical and horizontal spacing s [6], [17], [18], [24]. A Haar filter visual example is provided in Figure 13.

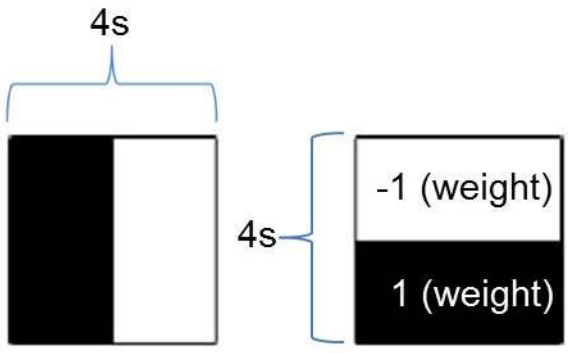

Figure 13. Haar box filters used for SURF, after [24].

The left and right Haar filters shown in Figure 13 are used to determine the responses in the x and y directions, respectively [6], [17], [18]. When the integral image method for box filtering discussed in Section II.A.2.a(3) is used to compute the response at each sample, the Haar convolution filters take only six operations to compute [24]. The resulting responses have a Gaussian weight applied that is centered at the interest point with a $2.5s$ standard deviation. The original SURF algorithm called for $2s$, but the OpenSURF version used here has a standard deviation value equal to $2.5s$ [6], [17], [18], [24]. After the responses are plotted on the (x, y) plane according to their filter responses in the corresponding direction, a 60° region (by SURF algorithm convention [18], [24]) is rotated around the origin, and the responses that fall within that area are summed [18]. The angle with the highest value of summed responses is selected as the orientation [24]. A visual display of this process is shown in Figure 14.

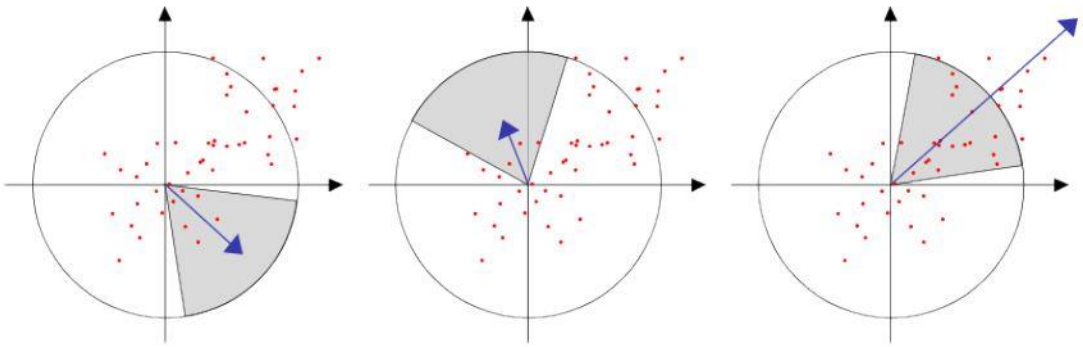

Figure 14. Orientation search and determination example for the SURF algorithm, from [24].

An example illustrating the orientation process followed within the search area surrounding the interest point is highlighted in Figure 15 with the Haar-wavelet kernel being applied.

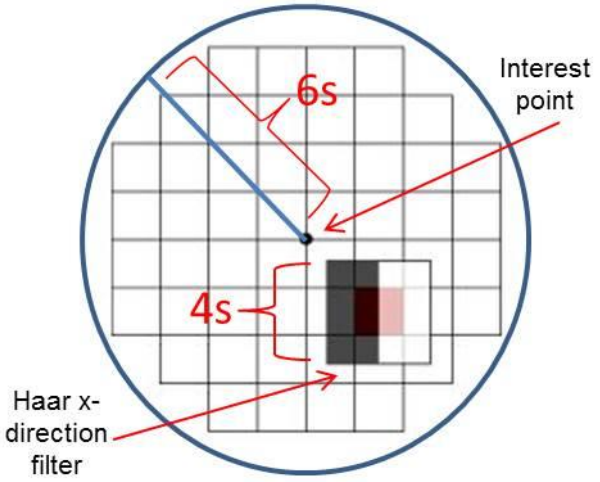

Figure 15. Example of Haar filtering during orientation assignment, after [18].

To understand what is measured by the Haar wavelet filters, refer to the work conducted on object detection by Papageorgiou et al. [30]. The simple wavelets used in SURF encode the differences in average intensities between regions [30]. Therefore, for the Haar kernels that measure responses in the x direction, a high response (magnitude) occurs if there is a change between pixel intensities in the corresponding direction of either increasing or decreasing magnitude. For high frequencies, the sum of absolute values is large. Visual examples illustrating these concepts are presented in Figure 18.

At this point in the algorithm, the orientation parameter has been calculated. So far, the interpolated (x, y) coordinates, the Fast Hessian determinant scale s, and the feature orientation represent the cumulative knowledge obtained for each feature. At this point, the algorithm's next step can be executed to determine the descriptor for each feature (or the region surrounding each interest point).

(2) *Description step two: computation of the sum of Haar wavelet descriptor.* The descriptor is used by the SURF algorithm to describe the properties of a feature that surrounds an interest point [6]. As previously stated, "blob-like" regions are detected by the Hessian-based detector, so the follow-on step is to define characteristics that describe those regions. Such description is obtained by drawing a square around the associated interest point and tilting it so the new *x*-axis lines up with the orientation line found in the algorithm's previous step [6], [17], [18].

The square regions' orientation around interest points and their associated orientation vectors are illustrated in Figure 16 on a graffiti image.

Figure 16. Example of SURF orientation and associated descriptor regions centered on interest points, from [17].

All square regions shown in Figure 16 have sides of length equal to $20s$, where s is (as before) the corresponding interest point's Fast Hessian determinant value [18]. Next, Haar-wavelet kernels are constructed for fast convolution filtering with side length equal to $2s$ [24]. The square region is then broken into 4×4 sub-regions [18], and each sub-region is assigned side length $5s$ (16 comprising the total square area) and convolutional filtering using Haar kernels is applied within each sub-region. Spatial information in the final descriptor is maintained by filtering the sub-regions independently [18]. Recall that the Haar filters and resulting responses are oriented along the new axes that were determined in the algorithm's orientation portion, so the new x' and y' directions now dictate the descriptor generation axis alignment. Note that x' is now aligned with the orientation vector and y' is perpendicular to x' [18]. The resulting responses from each sub-region are weighted with a Gaussian of standard deviation equal to $3.3s$ to increase robustness to geometric deformations and localization errors [24].

Since the main region surrounding the interest point is divided into a 4×4 block of sub-regions, there are a total of 16 sub-regions for the area of interest surrounding a given interest point [18]. In addition to the two Haar filter wavelet responses being summed within each sub-region, the absolute value of each Haar wavelet filter sample is also computed for future summation [6], [17], [18]. This process leads to four summations within each sub-regions used to characterize patterns as shown in Figure 18. Since there are 16 sub-regions, the overall descriptor is a vector that contains $categories \times subregions = 4\times16 = 64$ different entries [6], [17], [18], [24]. An example is provided in Figure 17 that contains a generic region of interest where each sub-region is has two samples in each direction (four total).

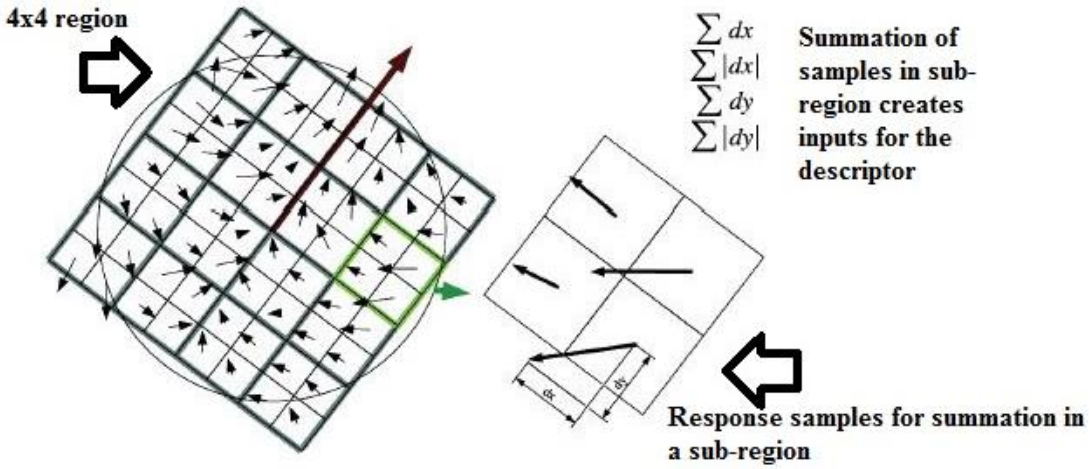

Figure 17. SURF descriptor generation example, after [18].

In Figure 17, all 16 sub-regions (in this case with four samples each) represent a spatial location within the area around the interest point. The four summation responses from each sub-region are placed in a consistent position in the descriptor vector, which allows the descriptor to carry the feature's spatial information [18]. The resulting descriptor vector associated with a given feature is generated from

$$SURF.\text{descriptor} = \left[\underbrace{\sum d_x, \sum |d_x|, \sum d_y, \sum |d_y|}_{subRegion1}, \underbrace{\sum d_x, \sum |d_x|, \sum d_y, \sum |d_y|}_{subRegion2}, \ldots, \underbrace{\sum d_x, \sum |d_x|, \sum d_y, \sum |d_y|}_{subRegion16} \right] \quad (7)$$

where the input to each entry in the descriptor is d_n, with n being the index [6], [17], [18]. The smaller square with four sample points shown in Figure 17 is an example of a sub-region.

In order to understand how this approach effectively defines common patterns, the SURF algorithm Haar response summation values for a few sample pixel patterns are shown in Figure 18.

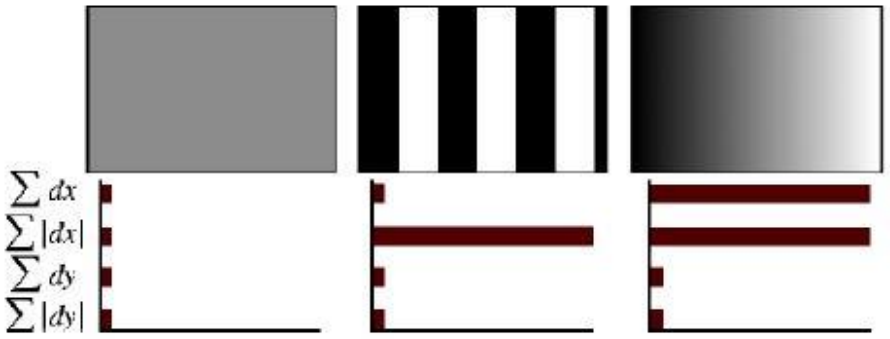

Figure 18. Example of Haar filters wavelet responses, from [18].

For example, high spatial frequency components lead to large values obtained when summing Haar response absolute values and small values for the regular summations.

In contrast, a gradual change in intensity along the x or y axis corresponds to an increase in both summations in the corresponding axis (again refer to Figure 18). The four summed values allow for feature characteristic differentiation within the sub-region. Note that if the pixel intensities are inverted with the same pattern (replace dark and light,

and vice versa), the sum of responses value is the opposite polarity with no sum of absolute values response change.

c. Sign of Laplacian Descriptor

Bay proposed calculating the Laplacian sign (polarity) for each interest point as a low-cost approach to increase speed for later fast indexing and matching [17], [18]. Since the Laplacian sign can indicate a blob that is darker or lighter than the background, it can also be used for quick matching [17], [18]. Bay proposed to use the sign of the Laplacian in k-dimensional (k-d) trees as a meaningful way to split the data [17]. This value is calculated at the same point in the algorithm as the Fast Hessian determinant and can be used for a possible matching algorithm [25].

d. SURF Algorithm Summary

The main steps present in the SURF algorithm are given by [6], [17], [18], [24], [25]:

- converting intensity image to grayscale,
- computing integral image,
- creating octaves and scale levels for scale space representation consisting of:

1. fast box filtering to create Gaussian second-order derivative approximations applied to the image,
2. increasing second-derivative approximation filter sizes to create scale levels,
3. computing Hessian determinant for each sampled pixel from the three box filter responses to be used in the scale DOG approximations in the octaves,
4. saving the sign of the Laplacian for each interest point for future use,
5. using four filter sizes to create a single octave's scales, with the next octave utilizing the second and fourth filter size in the previous for its first two scale levels,
6. downsampling each successive octave by two,

- conducting a 3D search for interest points in scale space and using NMS,
- determining sub-pixel accuracy location of interest points by quadratic interpolation,
- determining feature orientation using Haar wavelet response dominant direction, and
- generating descriptor on axis aligned to the feature orientation using Haar wavelet responses and their absolute values summed within 16 sub-regions [6], [17], [18] .

The SURF algorithm output includes a 64 length descriptor, (x, y) location, orientation angle, Fast Hessian determinant value (region size), and sign of Laplacian for each interest point [25].

e. Variations of SURF

Several SURF algorithm variations have been proposed, with two common ones listed below.

(1) Upright-SURF (U-SURF.) This is a version of SURF used for applications that do not require rotation invariance. This version removes the algorithm's orientation step. After calculating the Fast Hessian determinant and completing the 3D search, the algorithm moves directly into the interest point descriptor generation [6], [17], [18].

(2) SURF with varying size descriptors. This alternative allows for varying the size of a descriptor for either computational efficiency or to provide more feature characteristic details. In Section II.A.2.b(2), we saw that dividing the region of interest around the interest points into a 4×4 group of sections with four summed response values in each results in a 64 length descriptor. If the goal is lower resolution with higher matching speed, the interest area can be divided into a 3×3 group of sub-regions (for a total of nine), which gives a 36 length descriptor (nine times four) [6] [17] [18]. A commonly used descriptor with more detail is obtained in what is referred to as SURF-128 (128 length descriptor) [18], [25]. In order to generate twice as many feature entries as in the original algorithm, quantities $\sum_{dy<0} dx$ and $\sum_{dy<0} |dx|$ are computed separately from

$\sum_{dy \geq 0} dx$ and $\sum_{dy \geq 0} |dx|$. Similarly, the quantities $\sum_{dx<0} dy$ and $\sum_{dx<0} |dy|$ are computed separately from $\sum_{dx \geq 0} dy$ and $\sum_{dx \geq 0} |dy|$ [18], [25]. So, each of the 16 sub-regions has eight summations defined providing eight descriptor values leading to a descriptor of length $16 \times 8 = 128$ [18].

f. OpenSURF algorithm implementation

The baseline algorithm used for analysis in this thesis is a MATLAB implementation covered by the open-source BSD license [25]. The MATLAB version is a copy of the OpenSURF C# code by Evans and was downloaded from MathWorks [25]. This implementation has the option to choose a 64 or 128-size descriptor SURF and U-SURF [24], [25]. Additionally, it allows the user to select a threshold value for interest point detection, number of octaves, and initial sampling size used during the box-filtering step as well as a user-selected visual display of orientation Haar responses and the descriptor's responses for interest points [24], [25].

The output of this algorithm is contained in a MATLAB structured array *Ipts* with the interpolated coordinates (x, y), scale value from the Fast Hessian determinant, Laplacian, orientation angle (in radians), and descriptor for each interest point [25]. The algorithm comes with a display function that accepts the *Ipts* structured array and the original image and visually displays the SURF points with their location, scale level as circles, and orientation vector overlaid on the original image. The MATLAB OpenSURF code output is identical to the implementation proposed by Evans [24], [25].

The library implementation of SURF, as originally developed, is available for download but is closed-source and not available for commercial use [31]. There are a few open-source implementations including *OpenSURF* [25] and versions for *dlib* and *Pan-o-matic* [7]. Gossow et al. conducted performance testing [7] on the listed open-source versions and compared results to those obtained with the original SURF implementation [7]. The authors tested the performance in terms repeatability, recall, and precision as defined in [13], [14], [15]. The authors' comparison testing was based on the Mikolajczyk

et al. software framework [14], [15] (using the same testing process used by Bay et al. in the original SURF paper [17]) and utilized image sequences from [32]. Results showed the *Pan-o-matic* algorithm tracking almost exactly with the original SURF algorithm, followed by *OpenSURF* and then *dlib* [7]. Although the OpenSURF algorithm was slightly lower in repeatability for changes in viewpoint, blurring, dimming, zoom, and JPEG compression than that obtained from the original SURF implementation, the two implementation's results follow the same trend [7]. Since the algorithm performance is only slightly lower than that obtained from the original implementation, retains the same test result characteristics and is implemented open-source in MATLAB, OpenSURF fits the thesis goal to evaluate the modified algorithm performance [7], [25].

g. SURF Limitations

We described the different algorithm parts to include integral image, interest point detection in scale space, orientation assignment, and descriptor generation. One problem is that the algorithm is heavily dependent on the Fast Hessian determinant. This determinant value is used in the region size determination for both orientation and descriptor generation, the Haar filter kernel size, and in the Gaussian weighting functions [6], [17], [18]. If the detector Fast Hessian determinant value of this algorithm is not invariant to image characteristic changes, then the resulting orientation, descriptor, and matching algorithm may not necessarily be effective in matching features.

3. Comparisons of SURF and SIFT

Luo and Oubong compared the performance of SIFT, PCA-SIFT, and SURF using K-nearest neighbor and random sample consensus (RANSAC) for images that vary in scale, rotation, blur, illumination, and affine transformations [8].

Principal component analysis (PCA) SIFT, which the authors tested, is first introduced before discussing test results. The PCA-SIFT technique was developed by Yan and Sukthankar as an alternative way to determine the feature vector for the SIFT algorithm [33]. While the SIFT algorithm uses image gradient magnitude and orientation to create histograms in patches or sub-regions around the keypoint to develop the interest point descriptor, PCA-SIFT uses principal component analysis to create a descriptor.

Sub-pixel interest point determination, scale, and orientation are determined by following the same process as for original SIFT algorithm [33].

The first criterion measured was processing time using OpenCV algorithms2 [33]. The SURF algorithm greatly outperforms SIFT and PCA-SIFT by almost four orders of magnitude [33]. The results are listed in Table 3.

Table 3. SURF, SIFT, and PCA-SIFT algorithm speed comparison, after [8].

Items	SIFT	PCA-SIFT	SURF
Total matches	271	18	186
Total time (ms)	2.15378×10^{7}	2.13969×10^{7}	3362.86
10 match time (ms)	2.14806×10^{7}	2.09696×10^{7}	3304.97

Scale invariance tests show that the SIFT and SURF algorithms are comparable, while PCA-SIFT does not perform as well. The SURF algorithm performs poorly with regards to rotation invariance as can be seen in the data from Figure 19.

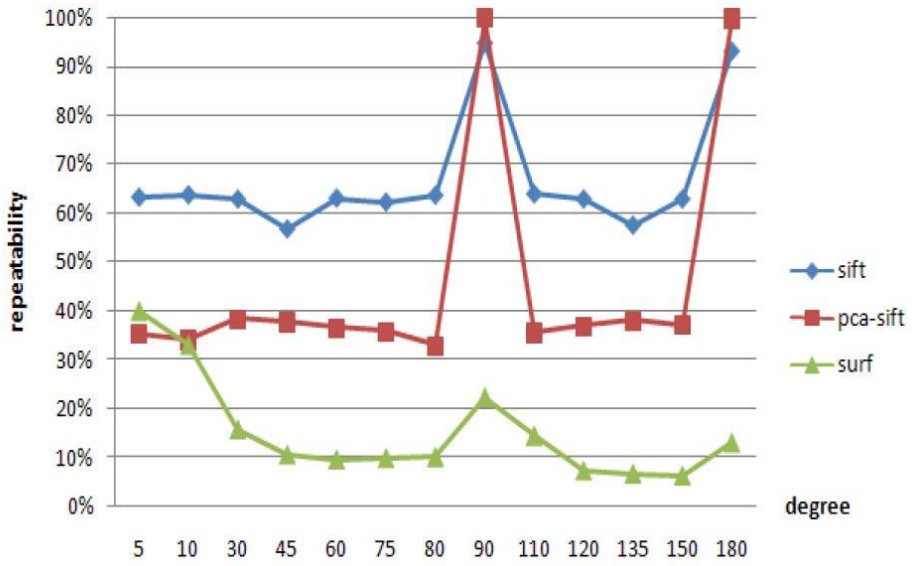

Figure 19. Test of SURF, SIFT and PCA-SIFT algorithm repeatability versus rotation, from [8].

SURF outperforms both in low to moderate image blurring, and also for illumination changes. For affine transformation (representing changes in viewpoint), the SURF algorithm performs the best for smaller changes, while the SIFT algorithm overtakes SURF for medium changes and PCA-SIFT for the largest changes [8]. A testing summary from Luo and Oubong's is shown in Table 4. The SURF algorithm appears to be more computationally efficient and robust to illumination changes. From the results in Figure 19, the authors discovered a SURF rotation invariance deficiency with improvement possible in scale, blur, and affine transformation invariance.

Table 4. A performance summary for invariance to image deformation for SURF, SIFT, and PCA-SIFT, from [8].

Method	Time	Scale	Rotation	Blur	Illumination	Affine
SIFT	common	best	best	best	common	good
PCA-SIFT	good	common	good	common	good	good
SURF	best	good	common	good	best	good

Comparing the SURF and SIFT algorithms with various descriptor sizes was also performed by Khan et al. on multiple larger benchmark datasets using a k-d tree for nearest-neighbor matching of individual features [9]. Their results conclude that the SURF and SIFT algorithms have similar performances for all tests with the exception of scaling, large blur, and viewpoint invariance [9]. The authors' proposed 64-dimension SIFT descriptor outperforms the original 128-dimension SIFT descriptor in both algorithm speed and viewpoint invariance [9].

Bauer et al. conducted another SURF and SIFT algorithm comparison test with both open-source and the original library implementations [10]. The authors tested for image transformations including rotation, scale, noise, illumination, and affine or viewpoint on outdoor and natural object images [10]. They used the criteria of total correct matches and the ratio of correct to incorrect matches as determined by a tool they wrote in Python [10]. Their conclusion was that SIFT has slightly better performance than SURF, but the results are very close for all tests. SIFT has a tendency to detect more interest points, which the authors saw as a computational disadvantage because there was

not a corresponding increase in correct matches. Overall, the authors concluded that SURF was a better choice given the computational efficiency and comparable performance [10].

4. Other Feature Detecting Algorithms

SURF and SIFT are commonly used to describe the complete detection and descriptor generation process; however, the SURF detector and SURF descriptor can function separately as long as the needed scale information is passed to the SURF descriptor algorithm from another detection algorithm. The following algorithms may include the detector-descriptor combination or just a detector. The function performed is specified for each algorithm. Note that we test these algorithms in this section as we are only testing SURF and the modified versions of SURF; however, these alternatives can also be used and are utilized for comparisons made in Chapters VI and VIII.

a. Harris Corner Detector

The Harris corner detector, proposed by Harris and Stephens in 1988 [34], is a widely used feature detection algorithm that focuses on the detection of corner regions. The main application targeted was camera motion and tracking designed to address the limitations of edge detection [34]. The author's goal was to start with the Moravec edge detector as the baseline, analyze its limitations, and design a more robust corner detection method [34].

The Moravec corner detection method computes the sum of square differences around a candidate pixel in various directions [34]. The Harris method uses the same approach but uses second-order derivative approximations to form a Hessian matrix estimate [34]. The difference of the Hessian determinant and square of the trace are computed at a given sample to characterize a corner region from

$$R = \det(H) - Tr(H)^2, \tag{8}$$

where H represents a Hessian matrix approximation [34]. If the resulting value of R is positive and a maximum with respect to the same calculation for its neighbors, then it is defined as a corner [34], [35].

b. Features and Accelerated Segment Test Corner Detector

The features and accelerated segment test (FAST) corner detection method was proposed by Rosten and Drummond to produce a computationally efficient approach compared to the SIFT and Harris corner detection algorithms for use on real-time video [35]. The FAST corner detector takes a circular region of 16 pixels (see Figure 20) surrounding a test candidate pixel and checks to determine if there is a set of n touching or adjacent pixels that are either of higher or lower intensity than the pixel of interest [35]. The number used for n can be any number, but the authors use $n = 9$ and $n = 12$. For faster computation, the algorithm first checks the four pixels on the outside circular region in Figure 20. The left, right, bottom, and top (labeled 1, 5, 9, 13) in relation to the candidate pixel are checked to determine if three of four have all higher intensity or all lower intensity than the candidate. If three of four conditions are met, then all pixels contained in the region are checked [35].

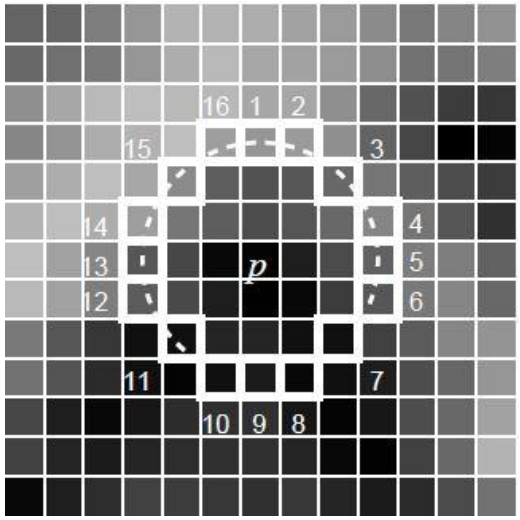

Figure 20. FAST image patch example used for corner detection, from [35].

The authors then use machine learning tools to develop a fast classifier [35]. In the authors' tests, they compare FAST with $n = 9$ and $n = 12$, the Harris corner detector, the Shi and Tomasi corner detection algorithm, the DOG used by Lowe for SIFT, and the SUSAN corner detector [35]. Results show the FAST detection with $n = 9$ outperforms

all other detectors in repeatability vs. corners per frame and speed on the four test sets used. The downside, as Rosten and Drummond point out, is that the FAST algorithm is sensitive to added noise [35].

c. Binary Robust Independent Elementary Features Descriptor

Calonder et al. proposed a feature descriptor called binary robust independent elementary features (BRIEF) that is binary for faster matching as an alternative to the efficiency techniques that had traditionally attempted to reduce descriptor length [36].

The authors take single pixel sample points contained within a region and apply smoothing kernels to reduce noise-sensitivity using a Gaussian kernel of size 9×9 [36]. The ultimate random sampling pattern was chosen after multiple tests were performed to determine the better resulting descriptor characteristics as shown in [36]. The authors then calculate an intensity comparison between selected pairs of points to determine the binary descriptor values. From this, they develop BRIEF-16, BRIEF-32, and BRIEF-64 algorithms, where the number after BRIEF represents the number of bytes needed to store the binary descriptor (corresponding to a descriptor size of 128, 256 and 512, respectively) [37].

BRIEF has a high sensitivity to feature rotation as it does not contain an orientation step in the algorithm; thus, the authors note that it should be compared to those obtained from U-SURF [36]. The authors do note that their algorithm outperforms SURF on small rotations in their single test, with the crossing point in recognition rate around 15° [36]. They use the same test set previously used by Bay in the original testing of SURF [18], [32], [32] (and the same used for this work) less the images test sets that vary orientation and scale. The authors conclude that the BRIEF-64 algorithm outperforms the SURF approach with the exception of the graffiti test set of Figure 33 and has significant speed increases in descriptor computation and matching when compared with SURF-64 [36]. The tests were conducted using the criterion of *recognition rate*, which is the number of correct matches divided by the total number of interest points detected by an algorithm between a pair of images [36].

d. Oriented FAST and ROTATED BRIEF

Rublee et al. produced an alternative to SURF and SIFT called oriented FAST and rotated BRIEF, with the goal of making a rotation invariant and noise resistant version of the BRIEF descriptor from Section II.A.4.c [38]. The authors used the FAST corner detector described in Section II.A.4.b to locate interest points with added orientation information in the algorithm [38].

After FAST interest point computation, the authors utilize a method of corner orientation by intensity centroid, which relies on computations of moments [38]. This algorithm assumes the bulk of the corner intensity not at the centroid, and the vector that defines the direction from the centroid to the intensity is utilized as an orientation. The centroid is quickly determined by four moments and the angle of orientation by two moments, with further details provided in [38]. From this orientation, the BRIEF descriptor is steered to the orientation with pre-computed lookup table of sample patterns in angular increments of 12° apart [38]. Last, to mitigate the loss of desired variance between sample pair comparisons in the steered version of BRIEF, they use a large training set to find the binary features that are uncorrelated and have high variance for their rotated lookup tables [38]. Variance is desired in a descriptor to provide characteristic uniqueness that allows for maximum differentiation between features [36], [38]. The detector-descriptor combination achieves a level of rotation invariance while still maintaining a speed advantage and comparable performance [38].

e. Binary Robust Invariant Scalable Keypoints

Binary robust invariant scalable keypoints (BRISK) was developed by Leutenegger et al. for use as a detector, descriptor, and descriptor matching algorithm [37]. The motivation behind BRISK was to push the usual tradeoff of detector and descriptor performance under image distortions versus computational speed [37]. The algorithm consists of two main stages: scale-space keypoint detection and keypoint descriptor generation followed by efficient matching due to a binary descriptor [37].

The first portion of the BRISK algorithm begins with scale space creation [37]. The BRISK algorithm then creates a predetermined number of octaves (usually four) by

downsampling the original image by two and then computing the same number of intra-octaves by first down-sampling by 1.5 to create the first intra-octave. After the first, following intra-octaves are downsampled by two rather than 1.5 [37]. The BRISK algorithm uses a FAST detector variant to create the values in the octaves used to find interest points [37]. A NMS and interest point sub-pixel interpolation is also applied [37]. The algorithm calculates object orientation around the interest point and then constructs a descriptor from uniform radius samples around an interest point, each with Gaussian smoothing applied [37]. The binary descriptor generation is accomplished in a similar way as for BRIEF with pairwise sample comparisons, but the BRISK algorithm uses fewer samples with deterministic spacing to reduce complexity with more pairwise comparisons than sample points [37]. The descriptor is 512 bits in length and binary, which matches the BRIEF-64 descriptor size [36], [37]. A BRISK sampling pattern example is seen in Figure 21.

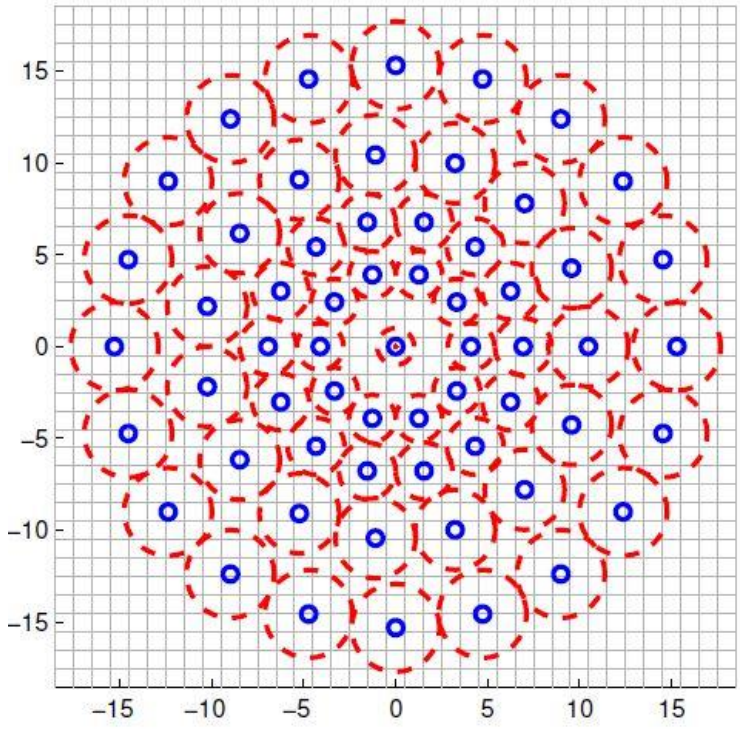

Figure 21. BRISK sampling pattern with 60 sample points, from [37].

The authors of the BRISK algorithm compared performance with the SURF algorithm, which was obtained by using the Mikolajczyk et al. testing framework for both detector and descriptor performance (adding SIFT just for the detector tests) [14], [15], [37]. Results show that the BRISK algorithm provides comparable performance with an increase in interest point detection and matching times [37].

f. Fast Retina Keypoint

Fast retina keypoint (FREAK) is a keypoint detector-descriptor proposed by Alahi et al. The authors claim that FREAK is computationally faster than the BRISK, SURF, and SIFT descriptors while being more robust [39]. In this approach, the descriptor is designed based on the human retina's behavior. Alahi et al. use a circular search region around the interest point [39]. The interest region sampling pattern has the highest concentration of samples close to the interest point with fewer the further away in the search area [39]. The regions around the samples are overlapping for better performance and weighted with increasing size Gaussian kernels for smoothing as shown in Figure 22.

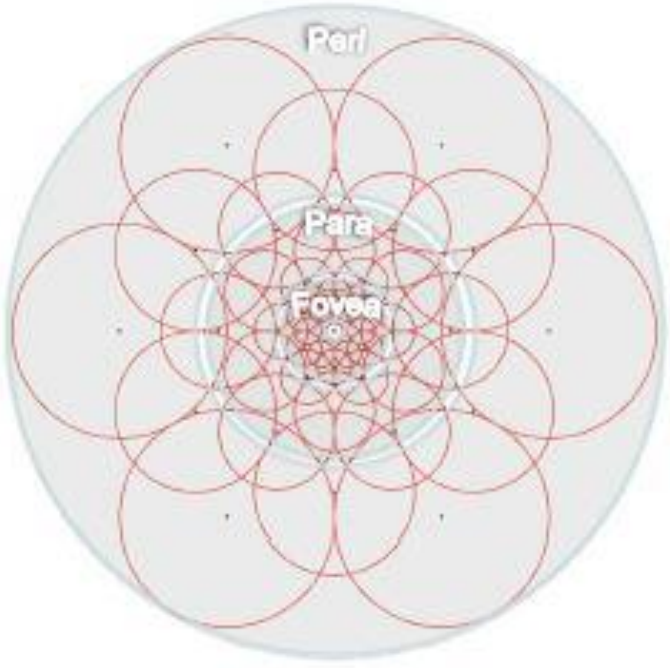

Figure 22. FREAK example of sampling with weighted Gaussian around interest point as related to the retina [39].

The FREAK descriptor is generated by comparing the smoothed intensity of pairs of sampled regions [39]. Instead of using distance between pairs as a criterion to compare two points, the authors conducted a test of close to 50,000 keypoints and matched pairs of samples to find those with the most distinctive and least correlated pairs [39]. The result is a binary FREAK descriptor with ordered entries based on the sample's Gaussian smoothing kernel scale (coarse to fine) [39]. The authors used the adaptive and generic accelerated segment test (AGAST) detector utilized by BRISK and compared FREAK, BRISK, SURF, and SIFT with the recall versus 1-precision graphs [39]. Their results showed that FREAK outperformed the other algorithms in the presence of a number of perturbations such as viewpoint changes, rotation, scale, blur, JPEG compression, and illumination changes. The authors also concluded BRISK was the second best performing in all tests. The SURF and SIFT algorithms alternated depending on the test and the images used [39]. The algorithm speed comparison results obtained for SIFT, SURF, BRISK and FREAK as determined by the authors of FREAK are provided in Table 5.

Table 5. SIFT, SURF, BRISK, and FREAK descriptor generation and matching speeds for approximately 1500 detected keypoints on Intel duo single-core 2.2 GHz processor, from [39].

Time per keypoint	SIFT	SURF	BRISK	FREAK
Description in [ms]	2.5	1.4	0.031	0.018
Matching time in [ns]	1014	566	36	25

5. Interest Point Matching

Interest point matching is a key portion of any detection or classification algorithm. If the descriptors are too big or too complex, matching is costly and the algorithm slow. Conversely, performance suffers if descriptors are too small and do not contain enough descriptive information. Additionally, in order to develop more efficient and effective methods for recognition in cluttered backgrounds and complex images, there is a need to understand how feature descriptors are organized and grouped and what those parameters represent. At each stage, understanding the interest point detector and

feature extraction algorithm capabilities and limitations is essential to evaluating how well it applies to the given matching scheme. In this section, some commonly used matching schemes for image features are described.

a. Nearest Neighbor Matching

We briefly introduce nearest neighbor (NN), nearest neighbor distance ratio (NNDR), and kd-tree matching. Both NN and NNDR require a distance metric [13]. The most commonly used metric is the Euclidean distance, but use of weighted distance metrics or Mahalanobis distance is also an option [13].

In multi-dimensional space, the Euclidean distance is the sum of squares of the distance between dimensional entries [13]. So, for the 64-point SURF descriptor, the sum of squares of all 64 elements between two descriptors gives the Euclidean distance in feature space. In addition, there are multiple ways to weight the variables, resulting in weighted Euclidean distances [13].

For NN matching, two neighboring features match when there are no other neighbors with smaller distances and their distance metric falls below a predetermined threshold [13]. In NN matching, a feature can have at most one match [13].

An alternative to the NN approach is NNDR matching, which uses a distance ratio of pixel to its nearest neighbor divided by distance to the second nearest neighbor [13]. The point of interest and its nearest neighbor are considered a match when this ratio falls below a predetermined threshold [13].

The most commonly used algorithm for nearest neighbor searching is the kd-tree, or a multi-dimensional search tree [13]. This algorithm divides the dimensions along axis-aligned hyperplanes and organizes the tree to allow for faster search efficiency [13]. This algorithm works well in lower-dimensional searches but does not perform as well with increased dimensionality [40].

b. Threshold Matching

Threshold matching evaluates two descriptors, and descriptors are considered a match when the distance between them is below a predetermined threshold [13]. With

threshold matching, a descriptor can have several matches [14], which are not all correct. Additionally, the threshold is difficult to set properly and has varying usefulness depending on the feature space matching properties [13].

c. Bag of Visual Words Method

Sivic and Zisserman proposed this method to classify image feature descriptors into visual words [41]. Here the goal is to group features in a way that allows similar techniques commonly used in database word retrieval [41]. In order to build the "visual vocabulary" in the image recognition application, feature descriptors are first quantized into clusters that represent the visual words [41]. This step can be carried out by any clustering scheme. Sivic and Zisserman used K-means clustering for their bag of visual words (BOVW) algorithm [41]. These clusters of features are independently defined using aggregated covariance matrices [41].

Once the "visual" word database is constructed, words can be weighted by standard techniques. These weights help to give higher values to frequently used words associated with a particular image while giving a lower weight to words that often appear in the overall database across many unrelated images [41]. Weighting also allows visual words that are highly specific to the image to be assigned a greater value, making the descriptor more effective as a classifier.

The main benefit of creating a database that is organized in similar fashion to a word database is that it allows for rapid object or image matching [41]. A time-consuming part of image recognition is the matching stage so optimization can result in recognition process efficiency improvement.

d. Interest Point Groups

The purpose of this approach is for fast database searching similar to that available from the BOVW method, with the exception that spatial information is consolidated and retained [42]. Lei Yu et al. proposed a technique that takes existing interest points from a given feature extraction algorithm (SIFT, SURF, etc.) and grouped all interest points inside circular regions with radius equal to a dedicated interest point's

scale [42]. From those interest points, the scale values are divided into bins, and only the interest points in the same bin as the original one are retained. From these, a weighted BOVW algorithm is applied to classify the interest point group to a visual word [42].

Ng and Kingsbury used pairwise spatial constraints between images to match interest point groups [43]. The authors first took a group of interest points and formed a pairwise spatial relationships matrix between each [43]. They then compared the spatial relationship similarity between two groups and also compared the pairwise similarity of orientations and features between the two interest groups [43]. The authors tested their algorithm and evaluated it against SIFT. Results showed higher correspondence of matched pairs is obtained along with a 25% increase in computational time [43].

Additional work was carried out by Brown and Lowe on using interest point groups to develop invariant features [44]. The authors use geometrical methods from an interest point group in scale space to resample the image for new interest points along a determined plane [44]. The author's method rejects false matches using first the Hough transform and next a RANdom Sample Consensus method (discussed in the next section) [44]. In testing, the author's claim repeatability and accuracy are improved by this technique but conclude that future work should include more efficient implementations targeted towards making this method practical for use [44].

e. RANdom Sample Consensus

The RANdom Sample Consensus (RANSAC) algorithm was proposed by Fischer and Bolles [45]. The basic algorithm function is outlined below:

- Select random sample of minimum size required to fit model
- Compute a model from the sample set
- Compute the set of “inliers” from the data set for points that lie within a predetermined threshold of that model
- Repeat until model with the most inliers over all samples is discovered [45]

For SURF applications, the feature descriptors are the data set and the RANSAC algorithm attempts to make the best data model or curve fit. Outliers, which are

considered to be noise or objects of no interest, are discarded by this algorithm [45]; thus, the model described represents features that characterize the object of interest. Although a perfect model is the ideal case, in reality RANSAC is better suited to remove outliers from still backgrounds when faster moving objects are the target. The combination of RANSAC and SURF has been used successfully in video stabilization [46] but may not be as useful in specific image recognition applications where high clutter is present.

B. AREA OF INVESTIGATION

In order to better understand and improve the SURF algorithm functionality, we first analyzed how the algorithm steps worked to process images and generate interest points, orientations and descriptors.

1. SURF's Suitability as a Feature Detector

Ideally, a feature descriptor should be completely invariant to object position, orientation, scale, intensity, surroundings, partial occlusion, viewpoint (affine transforms), image compression, and operate in near real time. While it is impossible to fully achieve this in practice, the goal is to approach as closely as conceivable. Issues with the SURF algorithm were introduced in Sections II.A.2.g and II.A.3. Previous studies showed that SURF performs well relative to the other feature-based algorithms in speed, scale invariance, intensity changes, and object location but could use improvement in rotation invariance and large viewpoint changes [8], [9]. Additionally, enhancement in the ability to discriminate an object of interest from the background is desirable. Last, improvement in the invariance to all image transformations is a goal.

In Section II.A.5, methods that are commonly used to take groups of features and perform grouping or matching leading to classification were discussed. In order for the SURF features to be useful to those algorithms, they need the SURF detector-descriptor to be as invariant as possible to all changes that can occur between images. Although SURF is currently ranked near the top of comparison testing on feature detection techniques [7], [8], room for improving the algorithm to provide increased invariance to orientation and large affine transforms exists. As long as no compromising of algorithm efficiency or speed results from algorithm modification, a more robust feature extraction

algorithm results and is more useful and accurate in an efficient image recognition process.

2. Improvement in the Interest Point Detection

We focused effort on the SURF algorithm interest point detection. As previously described, the interest point detection and resulting Fast Hessian determinant scale are key to producing both the orientation and descriptor region. Although the algorithm is designed to detect "blob-like" structures, SURF was applied to basic shapes to determine performance in these detection scenarios. All tests were run in MATLAB with the OpenSURF algorithm. Five octaves, initial sample size two, and the default threshold is used for all tests unless otherwise specified.

a. Circular Shape

For the first basic shape, the SURF algorithm was applied to a basic circle image. The image was JPEG compressed of size equal to 570×572. The results of this algorithm is shown in Figure 23. The circular shape has SURF circle overlays centered at the interest point coordinates, with interest point strength represented by the circle size and a green orientation vector in each.

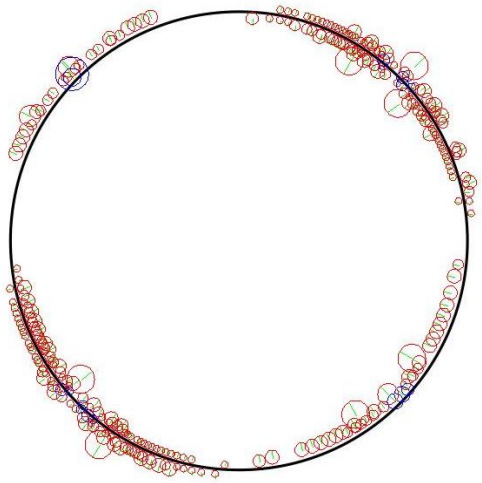

Figure 23. OpenSURF algorithm applied to a basic circle shape.

As previously stated, although the SURF algorithm is not designed to detect edges or corners specifically, the result is expected to be consistent regardless of the specific object's size and orientation. With the circle in Figure 23, it is seen that there are few interest points detected around the cardinal directions of rotation. Additionally, more interest points are detected located around the 45° and 225° directions as compared to those detected in the 135° and 315° directions. The inconsistent detection in these directions is due to the box filter orientation (specifically the D_{xy} filter). In computing the Fast Hessian determinant, the D_{yy} kernel detects circle's upper and lower parts, while the D_{xx} gets a zero response. The opposite is true for the left and right edges. The SURF filter responses for the individual kernels using filter size equal to nine are shown in Figure 24.

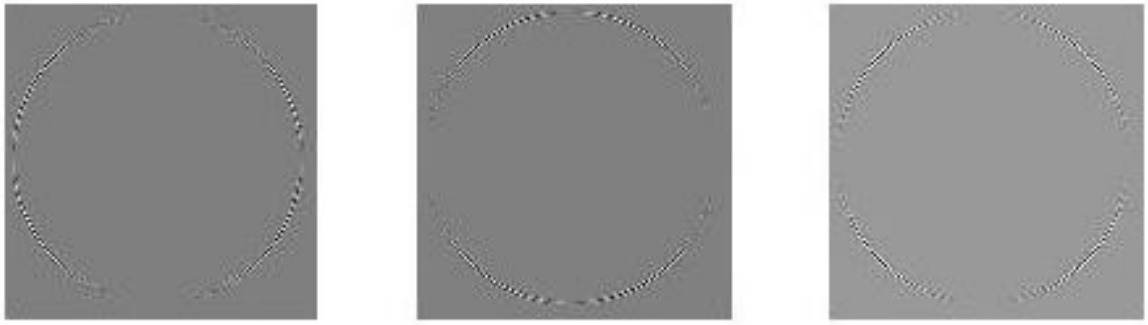

Figure 24. Fast Hessian box filter responses for circle image using a filter size equal to nine ($D_{xx} \rightarrow left, D_{yy} \rightarrow middle, D_{xy} \rightarrow right$).

It can be seen in Figure 24 that the D_{xx} box filter responses on the circle's top and bottom and the D_{yy} filter responses on left and right are weak. Since the Fast Hessian determinant multiplies D_{xx} and D_{yy} responses together, a small or zero response from either of these values results in their product being zero and the point of interest falling below the threshold in the interest point 3D search algorithm.

b. Hexagon Shape

Next, the SURF algorithm is applied to a hexagon shape to investigate the effect of corners with non-right angles and different orientations. There are also straight-edged sides with orientations along cardinal directions and ones that are offset. The hexagon in Figure 25 has JPEG compression and is of size equal to 276×250.

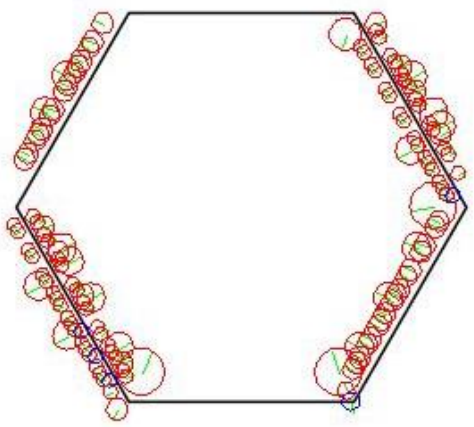

Figure 25. OpenSURF algorithm applied to a basic hexagon.

The SURF algorithm response obtained for the hexagon image shows the same inconsistencies as those observed for the circle, with non-vertical and horizontal edges receiving the majority of SURF points. In addition, there is also inconsistent detection for the corners.

c. Square Shape

We now move to a square that has four corners that ideally receive the same SURF points and four straight edges that are aligned with the cardinal directions. The square image in Figure 26 has JPEG compression and is of size equal to 751×722.

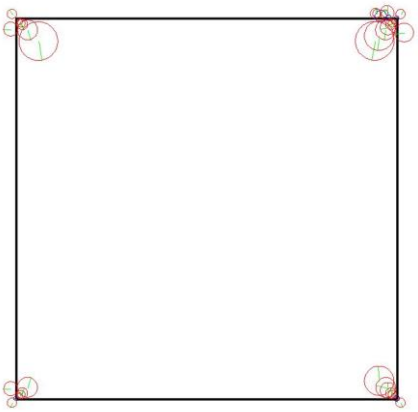

Figure 26. OpenSURF applied to a basic square.

Results shown in Figure 26 indicate that the detector ignores edges for the same reasons as noted for the circle. The algorithm is more consistent for corner detection on the square, but the number of interest points, size, and orientation are not identical for all.

d. Star Shape

The last basic shape considered is the star, which presents more complexity than the previous basic shapes considered. The star image in Figure 27 has JPEG compression and is of size equal to 571×494.

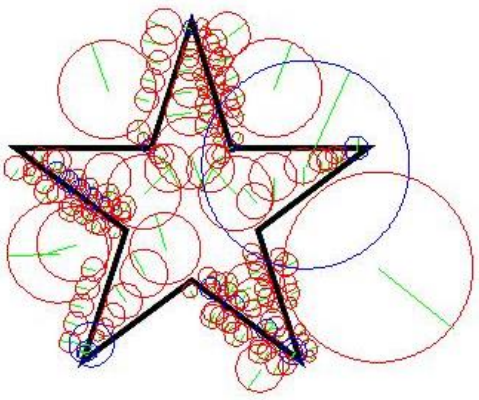

Figure 27. OpenSURF algorithm applied to a basic star shape.

The star shape shown in Figure 27 results in the highest number of SURF points for the basic shape examples considered. This is expected because the star has the highest number and concentration of "blob-like" regions; however, inconsistencies in the SURF points are still observed. Note that some of the strongest SURF points are inconsistent with regards to detection, orientation, and region size.

The second-derivatives, which are approximated in the Fast Hessian, give strong responses to blobs and ridges at certain angles. One problem is that a function based on the Hessian determinant does not detect long structures when the second-derivative is small [15]. This result can be useful to discriminate between possible objects with more complex shapes and edges but only when the detector is consistent regardless of object orientation.

C. EXPLANATION BEHIND ALGORITHM MODIFICATION CHOICE

In taking a deep look at the SURF algorithm, which is considered a comparison benchmark for many new feature detection algorithms [10], [36], [37], [39] , some consistency issues were discovered that led to efforts focused on modifying the algorithm in this work. The main reason behind such proposed modification is to obtain a more consistent detection process and Fast Hessian determinant values for image features. More detail is covered in the following sections.

1. Fast Hessian Determinant Weakness to Image Characteristic Changes

As seen in Section II.C.1, the Fast Hessian determinant gives different values for similar features in an image which is translated, rotated, or subject to other common transformations. This is apparent in the inconsistent detection results obtained for the similar features contained in the basic shapes shown in Section II.B.2.

In SURF, the algorithm's orientation step is designed to compensate for feature rotation effect by determining the Haar wavelet filter alignment for use in the region surrounding the interest point [6] by aligning the x response filter with the orientation vector. Since the orientation search region size, Haar filters size, and Gaussian smoothing kernels standard deviation are dependent on the Hessian determinant value, it is

important that consistent values be obtained under various image orientations and transformations. Additionally, the feature extraction results might not be reliable when some lines or edges are not properly or consistently detected.

2. Generating a Fast Algorithm Modification that Improves SURF Image Transformation Invariance

From Section II.A.3, it was seen that SURF compares favorably with other feature detection algorithms. This is especially true when accounting for algorithm speed versus the SIFT algorithm [8], [9], [10]. Although there are very slight advantages in performance to SIFT, the results for scale, illumination, compression, and blurring invariance for SURF are all very close to the best [7], [8], [9], [10]. Due to the fast convolution method of box filtering [6], [17], [18], SURF is consistently a faster algorithm than SIFT [8], [9] [10], with some programming implementations better optimized than others. This indicates that the SURF algorithm has a good basis for defining image features.

With the speed and effectiveness of SURF in mind, there are four main components to the algorithm which in principle we could improve: scale space formation, 3D interest point search, orientation assignment, and feature descriptor generation [6], [17], [18]. The main benefit of attempting to advance the interest point search is speed, which is beyond the scope of this work. The algorithm's orientation and feature descriptor generation stages which are based on Haar wavelets do well in defining the feature region details. As was described in Section II.A.2, the actual region and other tools are dependent on the Fast Hessian determinant value. Using integral images and box filters for fast convolution filtering is an extremely efficient approach to generate scale space difference of Gaussian approximations [6], [21], so this speed is an algorithm property that is desirable to maintain. Thus, what is left is integral image box filter modification. Therefore, improving the repeatability of the Fast Hessian determinants is focused on in this work, and this focus leads to the development of a possible solution for the SURF algorithm interest point detection limitations, which are discussed in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

III. OPENSURF ALGORITHM DEVELOPMENT AND TESTING METHODOLOGY

In this chapter we present a SURF algorithm variation which makes detected features less sensitive to image transformations. As seen in the previous chapters, the box filters used for the standard SURF implementation are not omnidirectional and tend to favor certain directions based on their static alignment with the image. We extend this class of filters by changing the "box" orientation by a 45° counterclockwise rotation.

A. DEVELOPMENT, MODIFICATION, AND USE OF ALGORITHMS

In this development, the algorithm described in Section II.A.2.f is utilized as the baseline for proposed modification. The algorithm chosen is covered under the Berkeley software distribution (BSD) 3-clause license, which allows for modification and redistribution as long as the license follows the code. It does not allow using the copyright owner's name to promote products and states that the code is provided as is [47]. An open-source license allows us the most flexibility for modification and future use. Note that a MATLAB testing environment does not provide the best results if speed is a required component of result for evaluation. However, MATLAB is a very effective tool in assessing the feature detector-descriptor performance in comparison to a baseline SURF implementation, which is the outcome desired for this work.

1. Modifications to the OpenSURF algorithm using MATLAB

We had two goals in modifying the OpenSURF algorithm to fit the testing needs. The first and primary one is to create a more robust interest point detector that is more invariant to rotation and other deformations than the original SURF. Second, a user-selected option is desired that provides the ability to visualize the integral image box filtering kernel responses for all of the different filter sizes. This option allows for a more detailed analysis as to why the algorithm detects certain features in some images but not others in order to give insight into the limitations and areas for improving the SURF algorithm.

2. Algorithm Additions for Troubleshooting and Visual Exploration

The OpenSURF algorithm has visual tools for both the orientation and descriptor for each interest point but lacks a visual tool to assess how the responses of each Fast Hessian filter affect interest point detection [25]. The *integralFilter* function from the MATLAB 2013b computer vision toolbox was utilized, with filters carefully generated (both rotated and regular) using the *integralKernel* function. A user-selected option to display the filter responses from the integral image for the D_{xx}, D_{yy}, and D_{xy} filters as well as the results from Equation (4) for one filter is provided. Additionally, the user can select to display the image response results of Equation (4) for all filter sizes. This option provides the opportunity to deeply analyze how individual filters affect the SURF algorithm interest point generation and gain insights as how to improve the feature detection portion.

For the single filter option, the algorithm also displays the box filter in a version that can be analyzed down to the pixel level. The 9×9 Fast Hessian box filter responses on the Lena image provided with the OpenSURF algorithm [25] is seen in Figure 28.

Figure 28. D_{xx} (left), D_{yy} (middle), and D_{xy} responses from SURF Fast Hessian box filtering with size 9×9 kernels.

3. Rotated SURF and Combined SURF Algorithm Additions

The primary method developed to improve consistency in the Fast Hessian determinant value to common rotation image changes was to use rotated box filters. The rotated filters can be designed in a fashion that is computationally as efficient as those in

the original SURF algorithm. The combined version requires twice as many calculations in the box filtering stage (though the combined version is optimal for parallel processing with no loss in computational efficiency). Additionally, time savings is achieved in reducing calculations associated with orientation, descriptor generation, and matching of undesired or noisy keypoints if the algorithm provides more discrimination in detecting only useful interest points.

a. Rotated SURF

The rotated SURF (R-SURF) development for the 45° rotated box filter implementation is presented in this section. Recall, three filters (named D_{xx}, D_{yy}, and D_{xy}) are used to generate the DOG approximations at different octaves in scale space. In order to maintain the proper second-derivative Gaussian standard deviation kernels that generate the approximated DOG within the octaves, care needs to go in to maintaining kernel proportions (length and width) that are close to those in the original Fast Hessian.

With these constraints in mind, rotated box filters were generated with a 45° counterclockwise rotation. Examples of size 15 filters are shown in Figure 30. The 45° amount of angle rotation was chosen for two reasons. The first is that Hessian-based detectors were shown weakest in their repeatability at odd multiples of 45° rotation between images [18]; thus, we hypothesized that responses more robust to image rotation and other transformations could be obtained if the box filter kernels were oriented in a direction of low repeatability observed in the original SURF algorithm. A second reason for choosing this rotation angle is that it corresponds to box filters that can be applied as efficiently as those used in the original SURF algorithm.

One SURF algorithm benefit is that it contains an extremely fast four-array access lookup for computing the box filter results of integral images as described in Sections II.A.2.a(2) and (3), and the speed advantage of four array lookups per box filter can be maintained if the SURF filters are rotated by 45°. Lienhart and Maydt proposed a four-index lookup for a 45° offset rectangle of pixels that have had the integral image operation performed [48]. From Figure 29, the four array indices required to obtain the

sum of pixel values for a rotated box region is seen. The formula for computing the sum of all image elements subjected to the rotated box filter is defined as

$$A = L_4 + L_1 - L_2 - L_3, \tag{9}$$

where A is the sum of the pixel values prior to integral image and L_n are the pixel values at indicated locations after the integral image computation is performed [49].

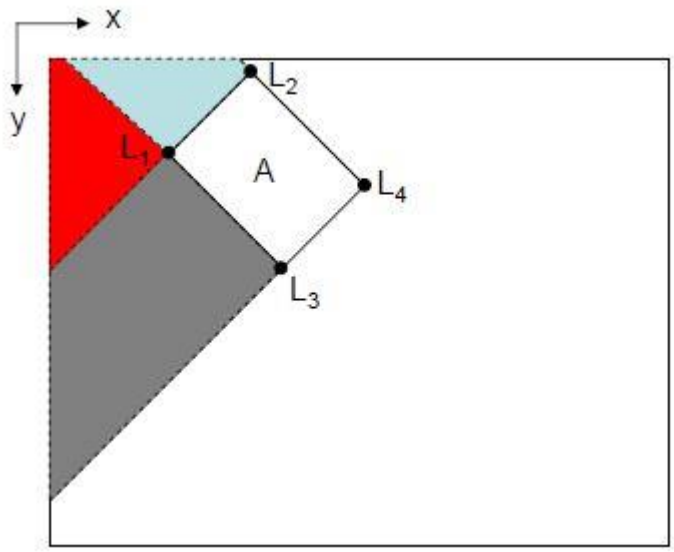

Figure 29. Rotated summed area table (RSAT) example [49].

Although the original second-derivative Gaussian filter kernel's ratio of length and width were not exactly maintained due to the difference in individual pixel diagonal corner to corner length (when compared that from the horizontal and vertical side to side), care was taken to approximate the filter kernel dimensions as closely as possible. Rotated Fast Hessian filter kernels of size equal to 15 are displayed in Figure 30.

b. Combined SURF

The combined SURF (C-SURF) algorithm is a simple modification using both the regular SURF Fast Hessian responses from each filter size and adding them together. Next, the sum is divided by two. Additionally, a stochastic version was developed where the regular to rotated variable ratio for the summation is determined by a uniform random

variable. The goal of this detector was to improve the rotation invariance, but from testing both the combined and combined stochastic detectors did not perform as well as the rotated and regular detectors. We assess that the rotated and regular filters are not as closely related as originally anticipated. This appears to result from either the filter kernel pixel structure or the rotated kernel length to width ratio not exactly matching that of the regular Fast Hessian filters. The pixel differences are most pronounced for the smaller filter sizes.

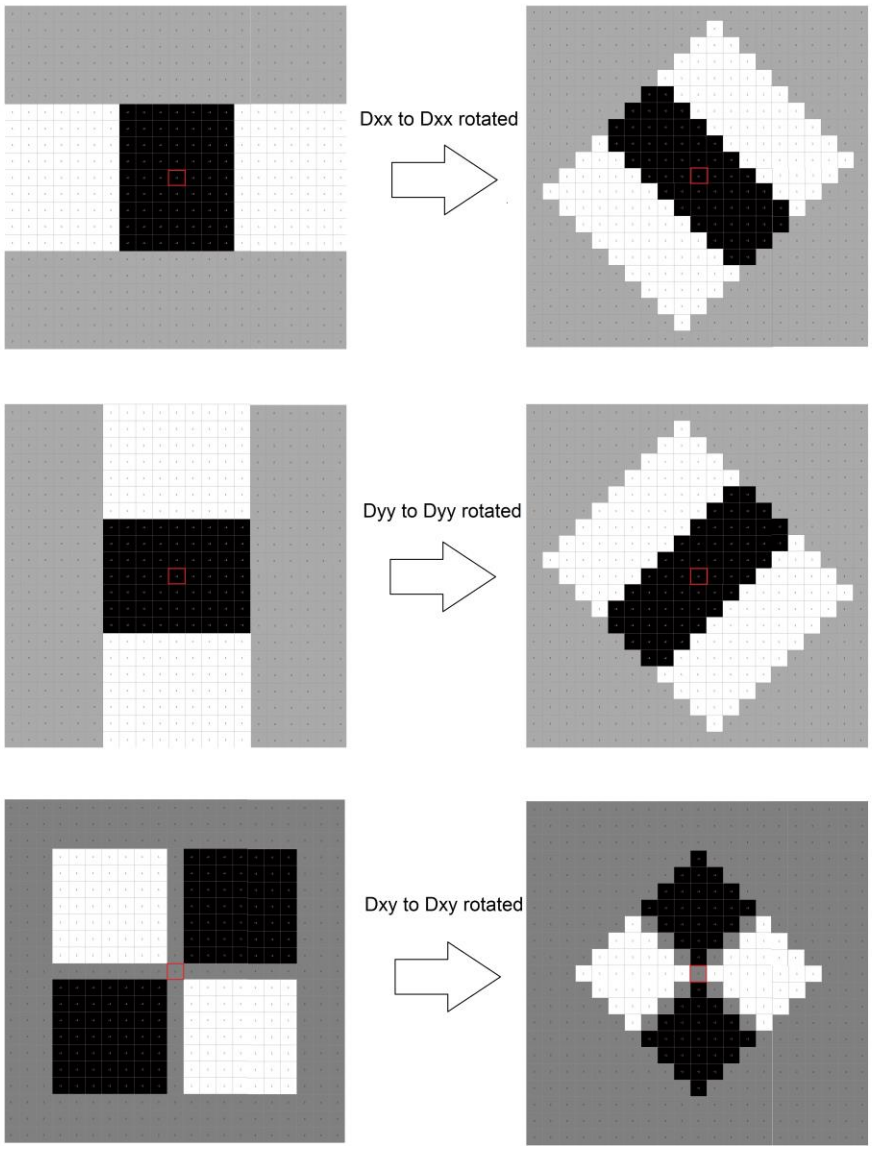

Figure 30. Rotated SURF box filter kernels for filter size 15 (Left side original SURF, right side rotated).

B. TESTING PLAN METHODOLOGY

The performance of the proposed rotated SURF algorithm was compared to that obtained with the original SURF 64-point descriptor algorithm with the same settings. The different image transformations presented below are tested:

- Affine (viewpoint)
- Blurring
- Illumination
- JPEG compression
- Scale with rotation

The test set was utilized by Mikolajczyk and Schmid and also Bay in his initial testing of the SURF algorithm [6], [17], [18]. The set consists of MATLAB test files obtained from the Katholieke Universiteit Leuven visual geometry group website where files are open source for use [32]. These tests use known homography between images displayed in Figure 32 through Figure 39. The homography is used as a testing algorithm input with the interest points and descriptor regions to investigate how well the detector-descriptor combinations performed through different image transformations [14], [15].

Note that the reason behind utilizing images that are homographically related is the ability to apply a matrix and the detector and descriptor results to a performance evaluation algorithm. Then, test results for detector region overlap and descriptor performance versus changes in viewpoint, rotational, or other image properties can be accurately obtained [15]. Mikolajczyz et al. reported their evaluation methods in two published papers: one for detectors and one for descriptors [14], [15]. A major goal in this thesis is detection testing on affine regions to measure the modified detector performance, so the detector testing methodology was selected [15]. *Repeatability* is one is a primary parameter used to evaluate a detector's performance, which is defined as the number of detected true positive (TP) correspondences between two regions under different geometric and image transformations [13] [15]. Another important factor is the location and follow-on descriptor region accuracy [15]. In this testing methodology,

repeatability is determined by a threshold value of overlap error ε_o between the resulting interest point and area resulting from the strength of the interest point. The threshold ε_o is applied to

$$1-\frac{area\left(R_{\mu_a}\cap R_{\left(H^T\mu_b H\right)}\right)}{area\left(R_{\mu_a}\cup R_{\left(H^T\mu_b H\right)}\right)}<\varepsilon_o, \tag{10}$$

where the variables a and b are indexes for the original and homographic-related image respectively. The elliptic region R_μ is defined by $x^T\mu x=1.$ The homography between the images H is a 3×3 matrix for each image pair in the test set [32]. *Repeatability* is then defined as the number of region-to-region correspondences that meet the threshold value of ε_o divided by the smaller number of total interest points between the two images [15]. The illustration in Figure 31 shows the effect overlap occurring between two regions has on the overlap error value.

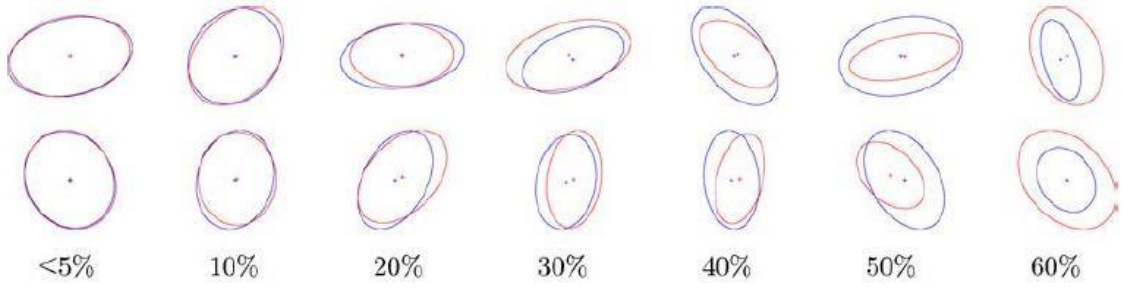

Figure 31. Examples of overlap error ε_o between corresponding regions, from [15].

A second criterion for detector evaluation is the *matching score*. A region match is defined as the number of correct detector matches where regions exceed the criteria $\varepsilon_o\le0.4$ or $40\%,$ with only a single match allowed per region. The *matching score* is the ratio of correct matches to the smaller number of detected points between the two images [15]. A match is determined by using the nearest neighbor concept in descriptor space with a Euclidean distance metric. The goal in this test is evaluate feature distinctiveness

[15]. While the *repeatability* focuses on the two region's location and overlap, the *matching score* metric takes into account the descriptor and how well it works with the detector to identify a matching region [15].

In addition to the differences in the image characteristics, the data set images are designed to presents a contrast between structured homogeneous scenes with distinctive edge boundaries and various types of textured scenes [14]. Each test set contains six images with different transformation levels applied, but for ease of viewing only four are displayed.

1. Affine (Viewpoint) Transformation Invariance

Affine invariance is another important characteristic for a feature detector and descriptor. An unpredictable factor for an incoming image is the viewpoint of a camera or sensor in relation to objects and features. For example, there is 360° of possible viewpoint range on land or sea and another 180° vertically in the case of maritime vessels. Viewpoint changes ranging from 20° to 60° were applied in both image sets from Figure 32 and Figure 33. Note, the brick wall is considered a textured scene while the graffiti is considered a structured scene [14].

Figure 32. Brick wall test set with affine (viewpoint) changes, after [32].

Figure 33. Graffiti wall test set with affine (viewpoint) changes, after [32].

2. Image Blurring Invariance

The sequence starts with the original image in the upper left of Figure 34 and Figure 35, and blurring distortion is progressively introduced. Five blurred versions are used in the tests in additional to the original image. Blurring invariance is important because it is usually not known whether the images received are from a clear day, one with high fog, a lot of rain, or from a camera with a dirty lens. The bikes present a structured scene, while the trees are a textured scene [14].

Figure 34. Bikes test set with varying of blurring, after [32].

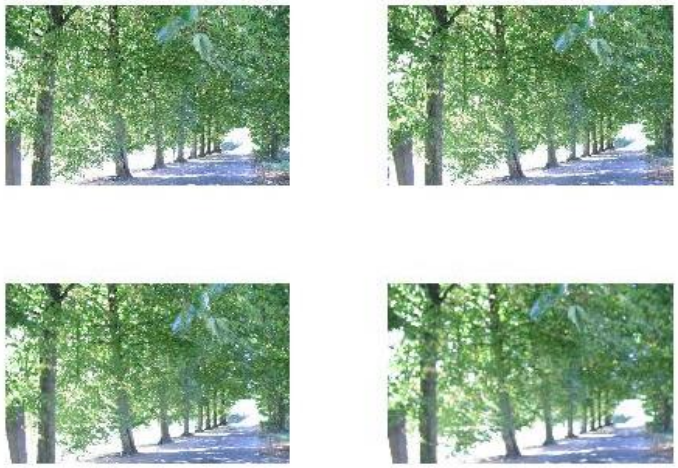

Figure 35. Trees test set with varying of blurring, after [32].

3. Illumination Change Invariance

Illumination change invariance is a critical part of a feature detection algorithm. A given object of interest can appear quite different when it is in the shade, recorded during a very bright and sunny day, or during dark and cloudy evening. The car and building data set shown in Figure 36 illustrate the potential changes in the image when camera settings are modified for the same image view, scale, and rotation. The car and building test set is considered a structured scene [14].

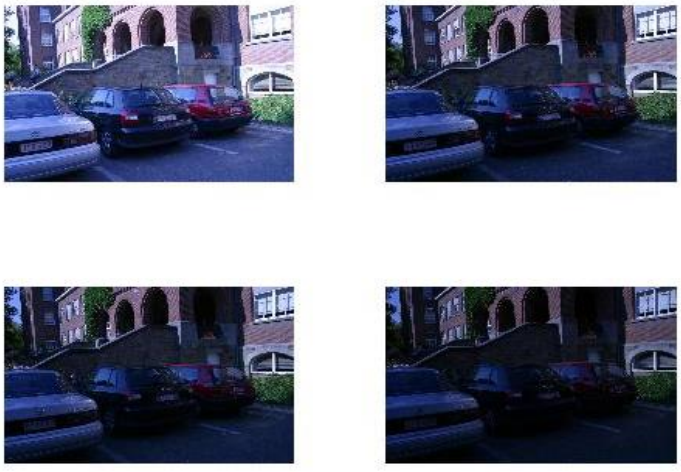

Figure 36. Cars and building test set with varying illumination levels, after [32].

4. Joint Photographic Experts Group (JPEG) Compression Invariance

JPEG Compression invariance is important in a feature detection algorithm. The impact resulting from increasing levels of JPEG compression is illustrated in Figure 37, where the original image shown at the top left and other images vary in compression up to over a 20-to-one compression ratio for the image contained in the bottom right of the figure. The detection and recognition of features should be robust to compression changes. The building images are considered structured scenes [14].

Figure 37. Building and trees test set with varying of JPEG compression, after [32].

5. Scale Invariance with Rotation

Next, two image sets used by Mikolajczyk and Schmid in their performance evaluation of multiple local descriptors are utilized. Both sets have six images and combine rotation and scale changes with rotation between 30° and 45° and scale changes between one and four. The bark scene in Figure 38 is a textured scene, and the boat scene in Figure 39 is a structured scene [14].

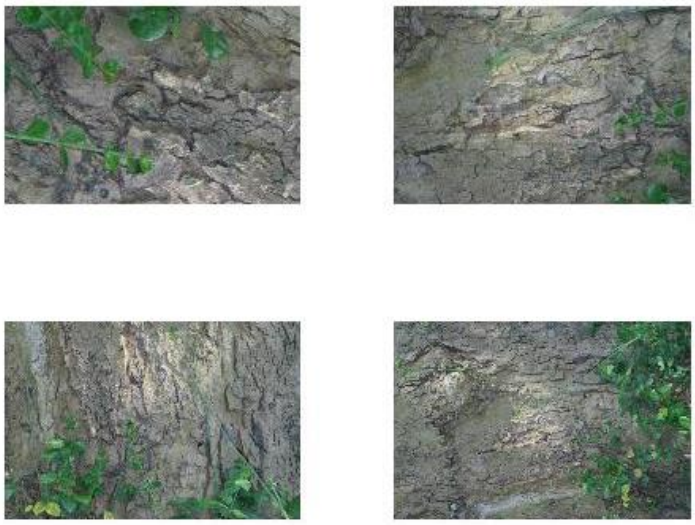

Figure 38. Bark and leaves test set with rotation and a scale factor change, after [32].

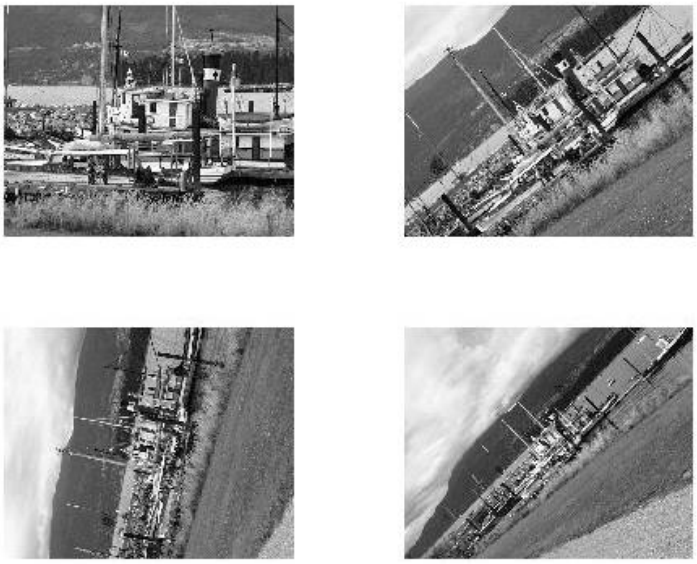

Figure 39. Boat test set with rotation and a scale factor change, after [32].

IV. SIMULATION RESULTS

The results obtained for the test scenarios discussed in Section III.B on all eight test sets are presented in this section [50]. All graphs in this section with the performance metric *repeatability* use a value for the overlap error $\varepsilon_o \leq 40\%$. More restrictive repeatability graphs with threshold value $\varepsilon_o \leq 20\%$ and less restrictive graphs with threshold value $\varepsilon_o \leq 60\%$ are provided in the Appendix for all detectors and test sets considered in this study. Recall the concept of overlap error ε_o, which was discussed in Section III.B and illustrated in Figure 31.

In all testing, identical settings were used for all OpenSURF algorithm detectors including the number of octaves (five), initial sampling size (two), and Fast Hessian interest point threshold value equal to 0.0002, which are default values for the original OpenSURF algorithm [25]. In addition to the original and rotate SURF version discussed earlier, a proposed combined algorithm was tested where the overall Fast Hessian determinant values used in that implementation is the averaged value obtained from the regular and rotated filters. Last, a proposed stochastic version of combined SURF algorithm was investigated. In the stochastic configuration, the two weighting parameters used to combine regular and rotated filters are randomly selected following a uniform density with values between 0 and 1. Specifically, the first coefficient is selected using the MATLAB *rand* function to weight each sample from the regular SURF filters, while $1-rand$ is applied to the rotated filter samples. The basic idea behind the stochastic methodology was to investigate whether a weighting other than basic averaging leads to better performance.

1. NNDR Matches to Total Ipts Ratio Test for all Test Sets

The first stage of our experiments investigated whether the rotated and combined algorithms performed well using a basic matching metric and test before moving into the testing to evaluate whether the matches were accurate. This test was to screen the detectors; if a detector is not able to provide matching criteria comparable to the original,

then it does not merit further testing. All eight image sets described in Section III.B were used for the first test. A MATLAB test function we created reshapes the *Ipts* structure data from all SURF algorithm outputs to a format that corresponds with the MATLAB *matchFeatures* function input requirements. This test function uses the NNDR matching approach and the Euclidian metric for the individual features. The cumulative results from the initial test in Figure 40 are displayed. Note that a larger value of NNDR is preferred; although, with this test it is not known whether the match is correct or incorrect. If the NNDR-to-Ipts ratio is equal to one, then every possible interest point is matched to another in the second image.

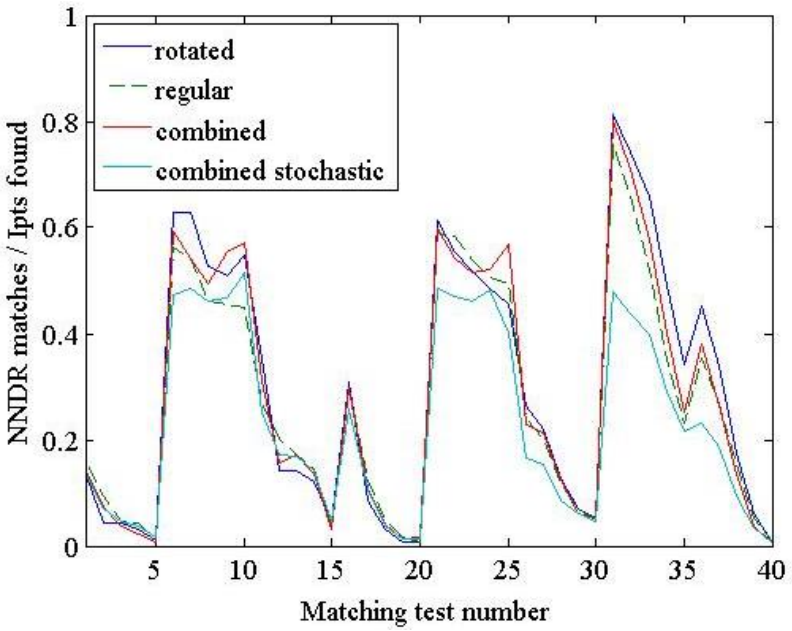

Figure 40. NNDR matches / total Ipts vs. test number for all eight affine covariant test sets.

Matching test numbers indicated in Figure 40 are associated to the test images as follows: one through five the bark and leaves test set, six through 10 the bikes, 11-15 the boat 16-20 the graffiti, 21-25 the cars and building, 26-30 the trees, 31-35 the building, and 26-40 the brick wall. All four detectors follow the same matching behavior patterns, with the combined stochastic method being outperformed consistently during this evaluation. Results show a large positive trend in the rotated algorithm's ability to

calculate a larger percentage of matches using a ratio of NNDR to the total number of *Ipts* found than the regular SURF algorithm for the bike, building, and brick wall test sets and comparable results for the others. Now that we have established that the new rotated detector achieves a higher percentage of detected point and descriptor matches for some of the tests, we investigate whether those matches are "quality" matches with the testing criteria discussed in Section III.B.

In the remaining tests, the *repeatability* graphs have an overlap error threshold ε_o less than 40%. This specific threshold value is that used by Mikolajczyz et al. in their detector testing comparison and is used here to provide a good baseline for reference [15]. Tests results obtained at other threshold values are included in the Appendix.

2. Affine Transformation Invariance

The detector's response to affine (viewpoint) variance was tested on the brick wall and graffiti data sets, respectively, shown in Figure 32 and Figure 33.

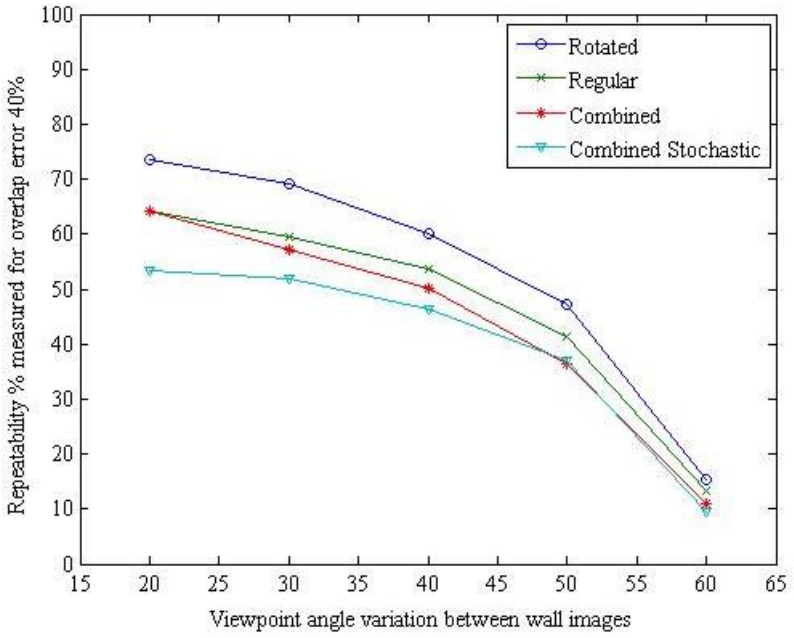

Figure 41. Repeatability versus viewpoint changes for tested detectors on brick wall test set with ε_o threshold value equal to 0.4.

Results indicate the rotated detector has the highest repeatability factor through the range of viewpoint changes considered for the textured brick wall scene, as shown in Figure 41. The regular detector starts out with around a 10% disadvantage in repeatability at a 20° viewpoint offset, with all detectors performing poorly at a large 60° viewpoint change value.

The same trend for matching scores as a function of viewpoint angle changes is observed in Figure 42 for the textured brick wall scene. The rotated detector performs better than the other detectors considered for viewpoint changes of 20° and 30°, with all detectors performing poorly at viewpoint changes larger than 40°.

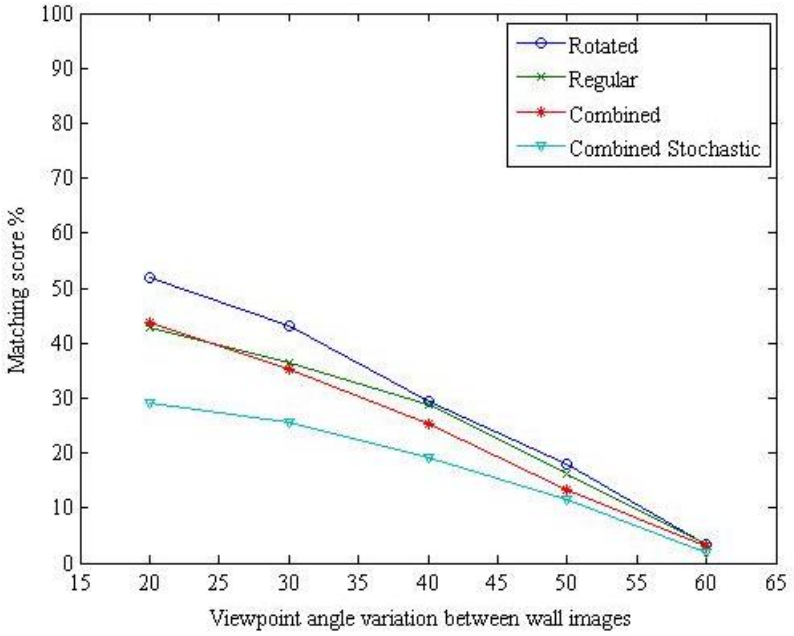

Figure 42. Matching score % versus viewpoint changes for tested detectors on brick wall test set.

Repeatability versus viewpoint changes for tested detectors on the brick wall test set with ε_o threshold value equal to 0.2 and 0.6 can be found in the Appendix in Figure 73 and Figure 74.

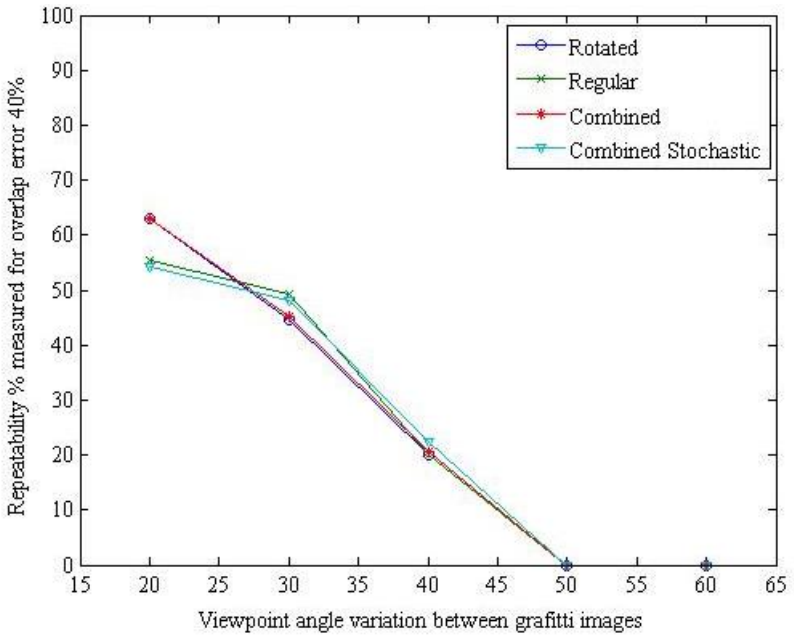

Figure 43. Repeatability versus viewpoint changes for tested detectors on graffiti test set with ε_o threshold value equal to 0.4.

The only notable takeaway point which can be made from the repeatability results obtained from the structured graffiti scene in Figure 43 is that the rotated and combined detectors perform better than the other two at a smaller viewpoint angle change of 20°. It is of note that the Fast Hessian based detectors all do very poorly at matching points on this textured scene at larger viewpoint changes. The results also show that the rotated detector performs better than the others considered when the overlap error threshold value ε_o increases to 60% as shown in Figure 66 of the Appendix.

For the graffiti test set, all detectors show poor repeatability scores as shown in Figure 44. With the low matching results, there is not much that can be derived from these results except that a Fast Hessian based detector does not appear to work well through affine transformations of a structured scene with the graffiti image properties.

Repeatability versus viewpoint changes for tested detectors on the graffiti test set with ε_o threshold value equal to 0.2 can be found in the Appendix in Figure 65 and Figure 66.

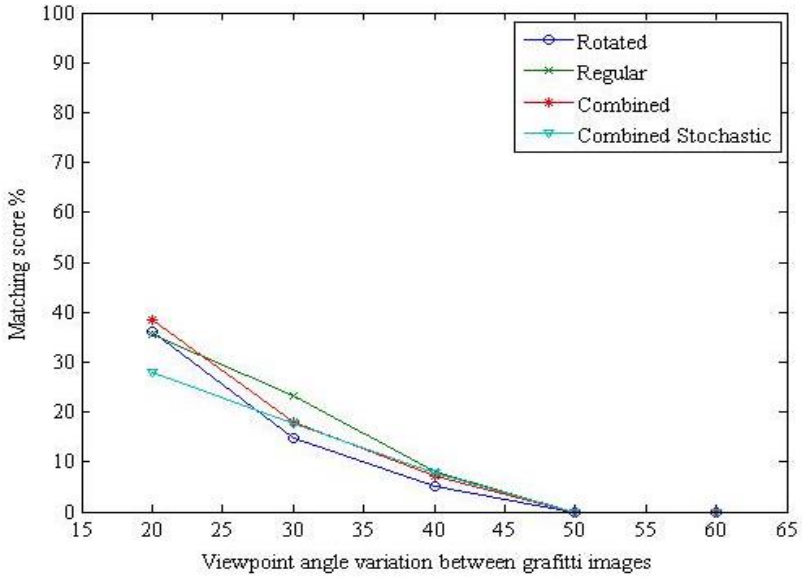

Figure 44. Matching score % versus viewpoint changes for tested detectors on graffiti test set.

3. Image Blurring Invariance

The detector response to blurring changes was tested using the bikes data set shown in Figure 34, and the trees test set shown in Figure 35.

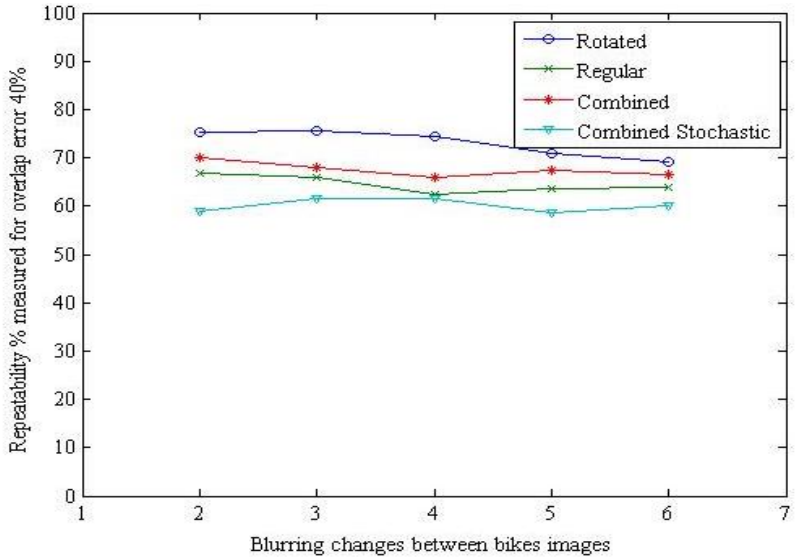

Figure 45. Repeatability versus blurring changes for tested detectors on bikes test set with ε_o threshold value equal to 0.4.

The results shown in Figure 45 indicate that the rotated detector performs the best for repeatability throughout the blurring range applied to the bikes image. The results in Figure 46 show the rotated detector has the higher matching scores than other detectors for the bikes test set. The regular and combined detectors track each other closely until the rotated detector degrades to their matching performance at higher blurring levels.

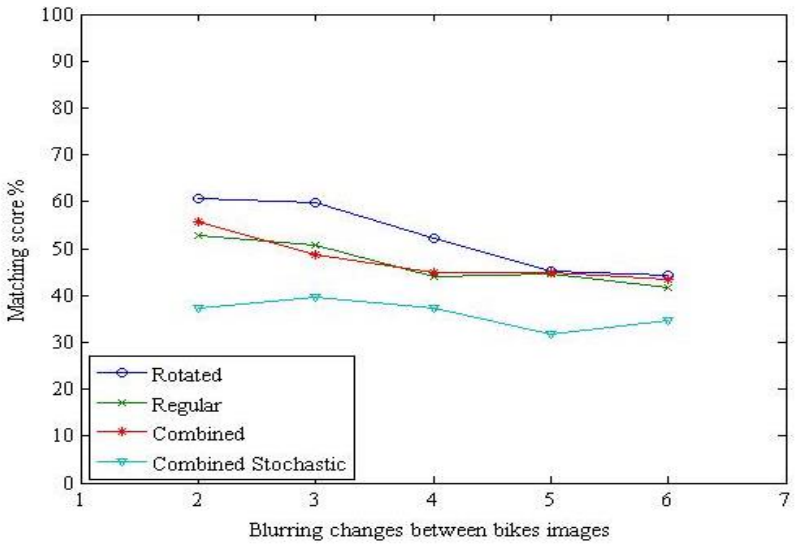

Figure 46. Matching score % versus blurring changes for tested detectors on bikes test set.

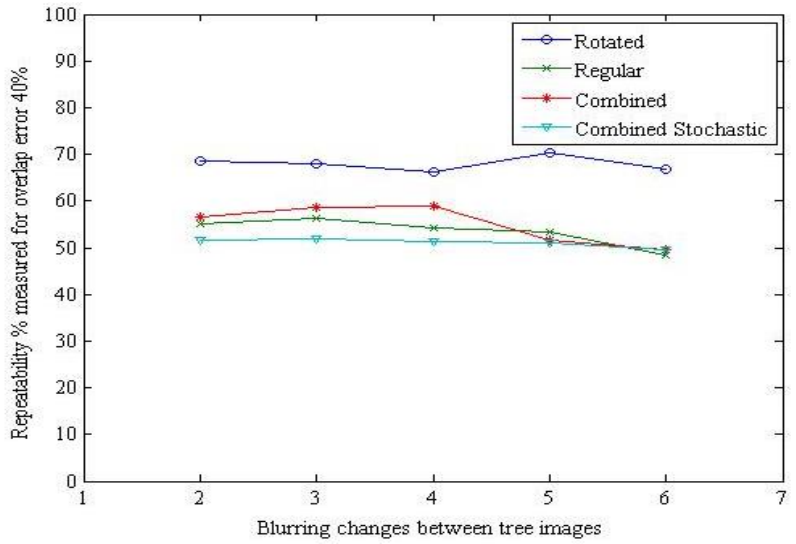

Figure 47. Repeatability versus blurring changes for tested detectors on trees test set with ε_o threshold value equal to 0.4.

Results obtained on the tree test set show the rotated detector has consistently higher repeatability results than the other detectors for all blurring levels considered, as shown in Figure 47.

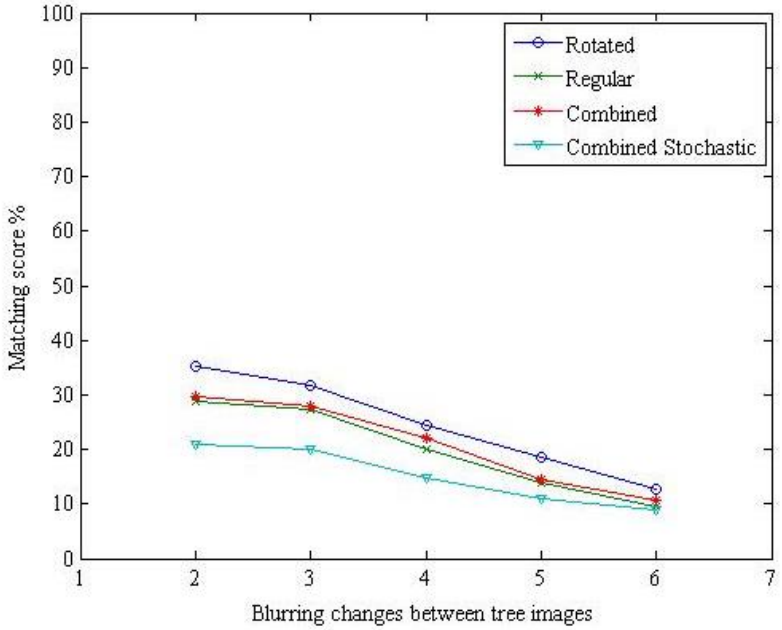

Figure 48. Matching score % versus blurring changes for tested detectors on trees test set.

Results shown in Figure 48 indicate the rotated detector has the highest matching scores throughout the blurring changes for the textured tree scene. Note the improvement is not as large as that observed for the repeatability tests shown in Figure 47. Further, the combined and regular detectors once again have very similar performances.

Repeatability versus viewpoint changes for tested detectors on the bikes and trees test sets with ε_o threshold value equal to 0.2 and 0.6 can be found in the Appendix in Figures 61, 62, 69, and 70.

4. Illumination Change Invariance

The detector response to illumination variance was tested using the cars and building data set shown in Figure 36.

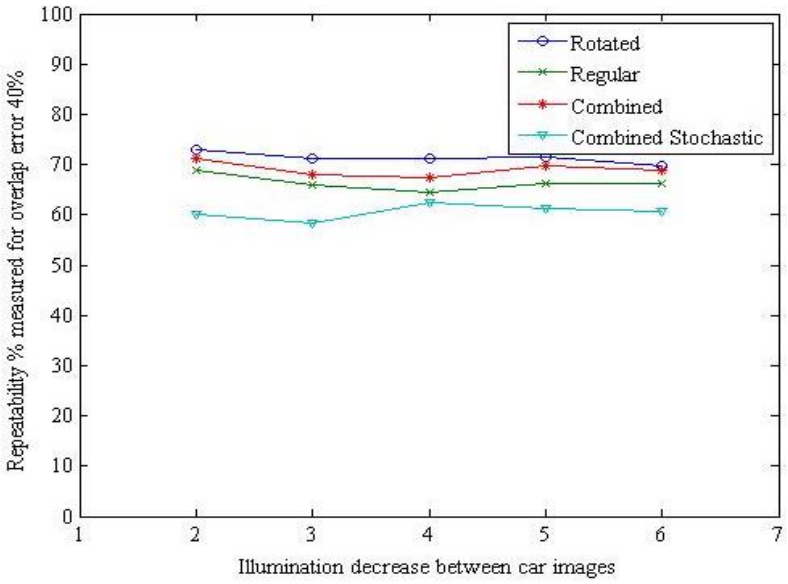

Figure 49. Repeatability versus illumination changes for tested detectors on cars and building test set with ε_o threshold value equal to 0.4.

Results from Figure 49 show the rotated detector having higher repeatability scores throughout the range of illumination changes applied to the structured car and building scene. This improvement is less than 5% over the combined detector throughout but ranges between 5 to 10% over performances obtained with the regular detector.

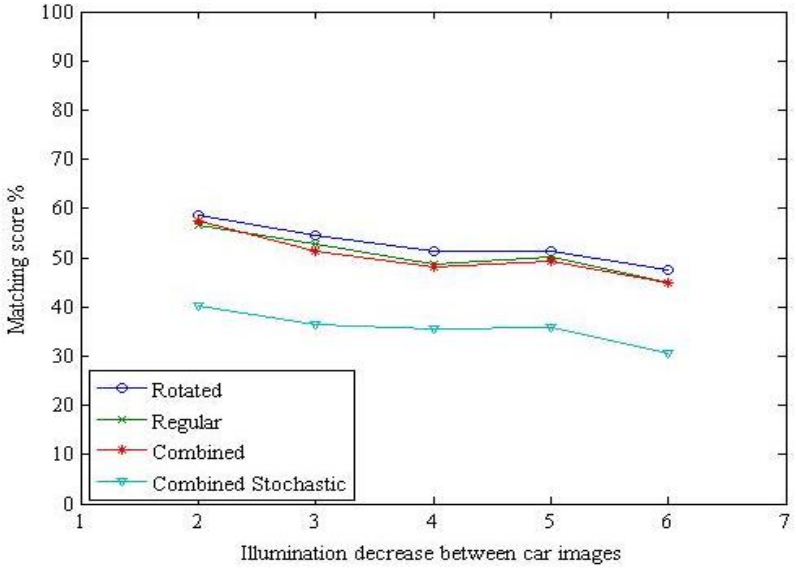

Figure 50. Matching score % versus illumination changes for tested detectors on cars and building test set.

The rotated detector again shows slightly better matching score performance throughout the range of illumination changes over the regular and combined detectors, as shown in Figure 50 for the structured car and building scene. Repeatability versus viewpoint changes for tested detectors on the cars and building test set with ε_o threshold value equal to 0.2 and 0.6 can be found in the Appendix in Figure 67 and Figure 68.

5. Joint Photographic Experts Group (JPEG) Compression Invariance

The detectors' response to JPEG compression variance was tested using the building data set in Figure 37.

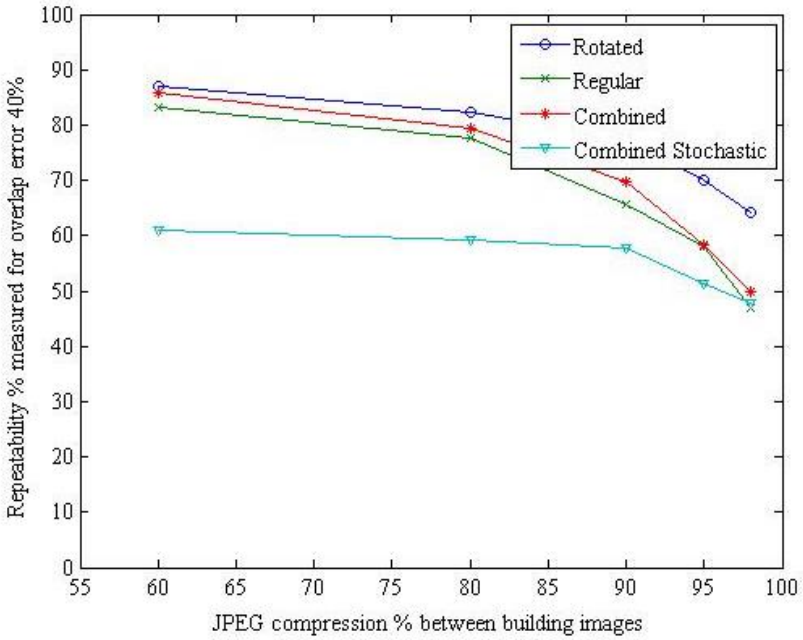

Figure 51. Repeatability versus JPEG compression changes for tested detectors on building test set with ε_o threshold value equal to 0.4.

Results show the rotated detector performs better with respect to repeatability throughout the range of JPEG compression levels considered as illustrated in Figure 51. The rotated detector appears to be more robust to increasing JPEG compression levels than the other three detectors considered. Finally, we note the rotated detector's matching score in Figure 52 follows the repeatability trends under JPEG compression variations for the structured building scene data set as those previously observed for repeatability.

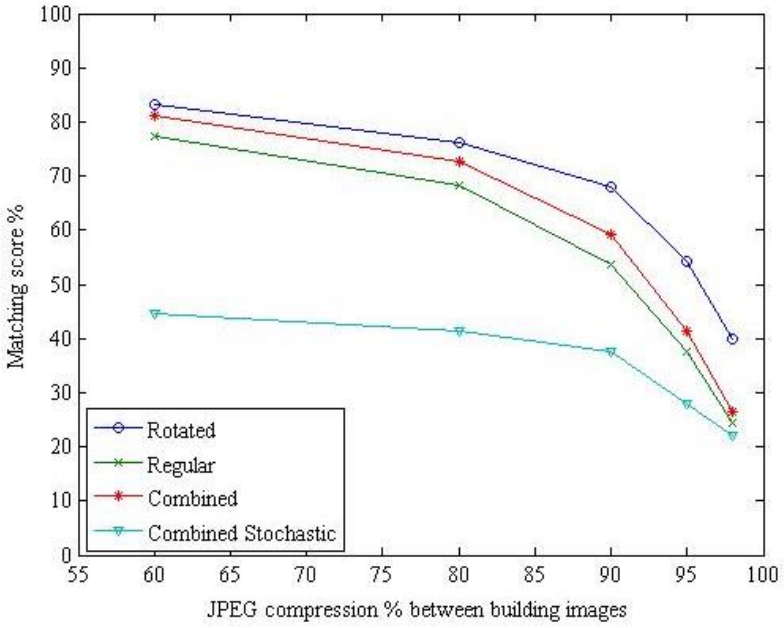

Figure 52. Matching score % versus JPEG compression changes for tested detectors on building test set.

6. Scale Invariance with Rotation

We utilized both the bark and leaves test set from Figure 38 and the boat test set from Figure 39 in testing for scale changes.

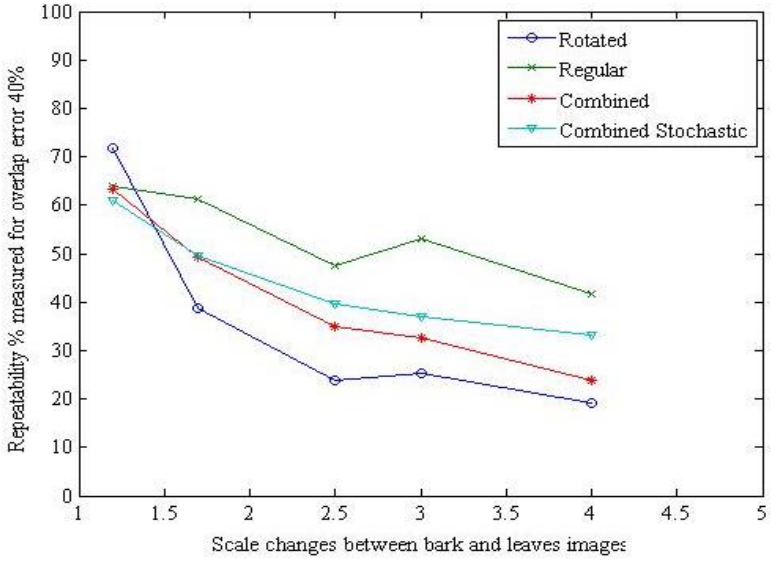

Figure 53. Repeatability versus scale changes for tested detectors on bark and leaves test set with ε_o threshold value equal to 0.4.

The results in Figure 53 show an initial higher repeatability for the rotated detector at small scale change value. However, the regular detector performs better for scale changes between 1.5 and two. Overall, the regular detector appears to be the best for this textured scene through transformations with larger scale changes and rotation.

The regular detector again outperforms in matching score results though the range of scale changes for the textured scene shown in Figure 54, with very close scores obtained for all detectors at the lower and higher end of scale changes.

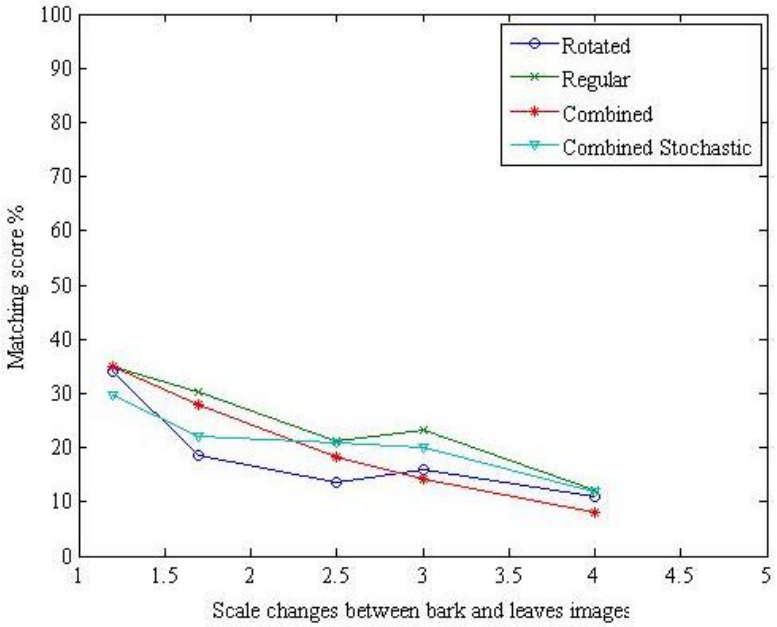

Figure 54. Matching score % versus scale changes for tested detectors on bark and leaves test set.

The rotated detector again performs slightly better than the other three detectors on small scale change values, while the regular detector performs slightly better for higher scale change values on the structured scene in Figure 55. Note that the rotated detector has the best results for scale changes on this structured scene with the overlap error threshold set to 60% in Figure 64 of the Appendix in Figures 59, 60, 63, and 64.

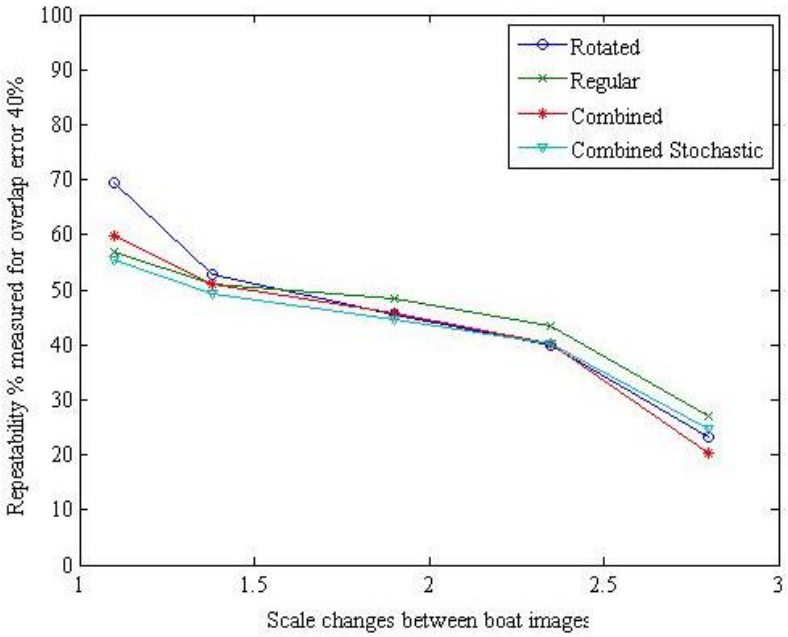

Figure 55. Repeatability versus scale changes for tested detectors on boat test set with ε_o threshold value equal to 0.4.

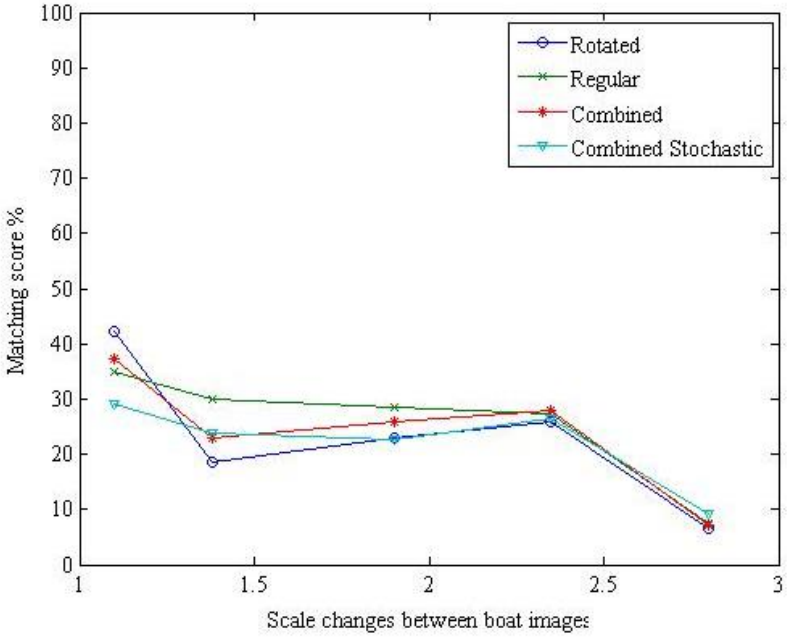

Figure 56. Matching score % versus scale changes for tested detectors on boat test set.

The rotated detector's matching score in Figure 56 is superior for smaller scale structured boat test set scene change values. The regular detector becomes the best performing for the second and third scale change increases, and all detectors converge for the two highest levels of scale change. Repeatability versus viewpoint changes for tested detectors on the boat and the bark and leaves test sets with ε_o threshold value equal to 0.2 and 0.6 can be found in the Appendix.

7. Results Summary

To summarize the results of this section, we state the following:

- The rotated and regular detectors both have performance advantages over each other dependent on the type of image transformation, transformation level, and scene characteristics.
- The combined and combined stochastic detectors appear to have weaker overall performance than the other two.
- The rotated detector is clearly better in the test results for image blurring, illumination changes and JPEG compression.
- The regular detector performs better for tests conducted when subjected to scale changes, though the rotated version is very good at low levels of this transformation.
- Since the test images in Figure 38 and Figure 39 exhibit both rotation in and scale change transformations, any future test should be designed to isolate the rotation and scale variation.
- For viewpoint or affine invariance the rotated detector clearly performs better on the texture brick wall scene of Figure 32.
- Finally, no detector performs well on larger viewpoint changes for the graffiti scene shown in Figure 33.

V. ORIGINAL WORK PERFORMED IN THIS THESIS

A. ALGORITHM DISPLAY AND TROUBLESHOOTING IMPROVEMENTS

A basic functional description for the modifications performed to enhance troubleshooting capability in the OpenSURF algorithm is found in Section III.A.1. The modified algorithm requires user selected inputs when the algorithm is initially run. The unmodified algorithm requires the user to input options as a structured array, which includes the number of octaves, threshold for interest point detection, initial 2D sampling size for the first octave, selection of a 64 or 128 point descriptor, choice of SURF or U-SURF versions, and a visual display of orientation responses and descriptor bins for each interest point. The user is given an option of either running the variations of SURF with default parameters (which defaults to the faster original algorithm), or selecting options using the pop-up shown in Figure 57.

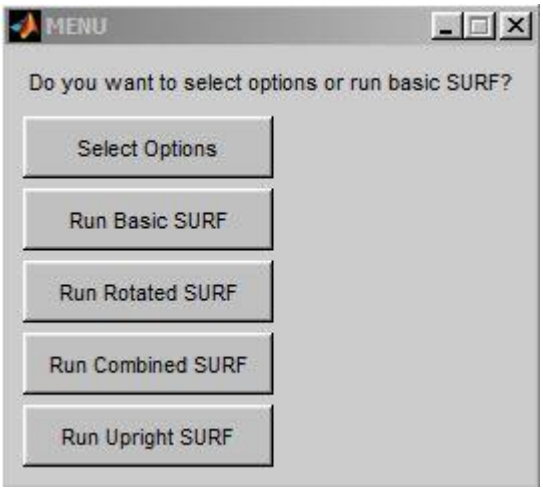

Figure 57. The initial OpenSURF selection menu addition.

When the "Select Options" button is chosen, a series of screens opens allowing the user to choose user-selected options as well as the option to generate visual responses for one or all filter sizes.

B. ROTATED SURF AND COMBINED SURF

The R-SURF and C-SURF options were added to the MATLAB OpenSURF implementation. A basic description of the modification's purpose was described in Section III.A.3. In order to implement these algorithms, the OpenSURF code was modified to include conditional statements based on the user-selected options to execute either the regular, rotated, or combined SURF options. Refer to Figure 58 for an updated algorithm flow-chart, which contains the R-SURF and C-SURF additions as well as the filter response options described in Section V.A.

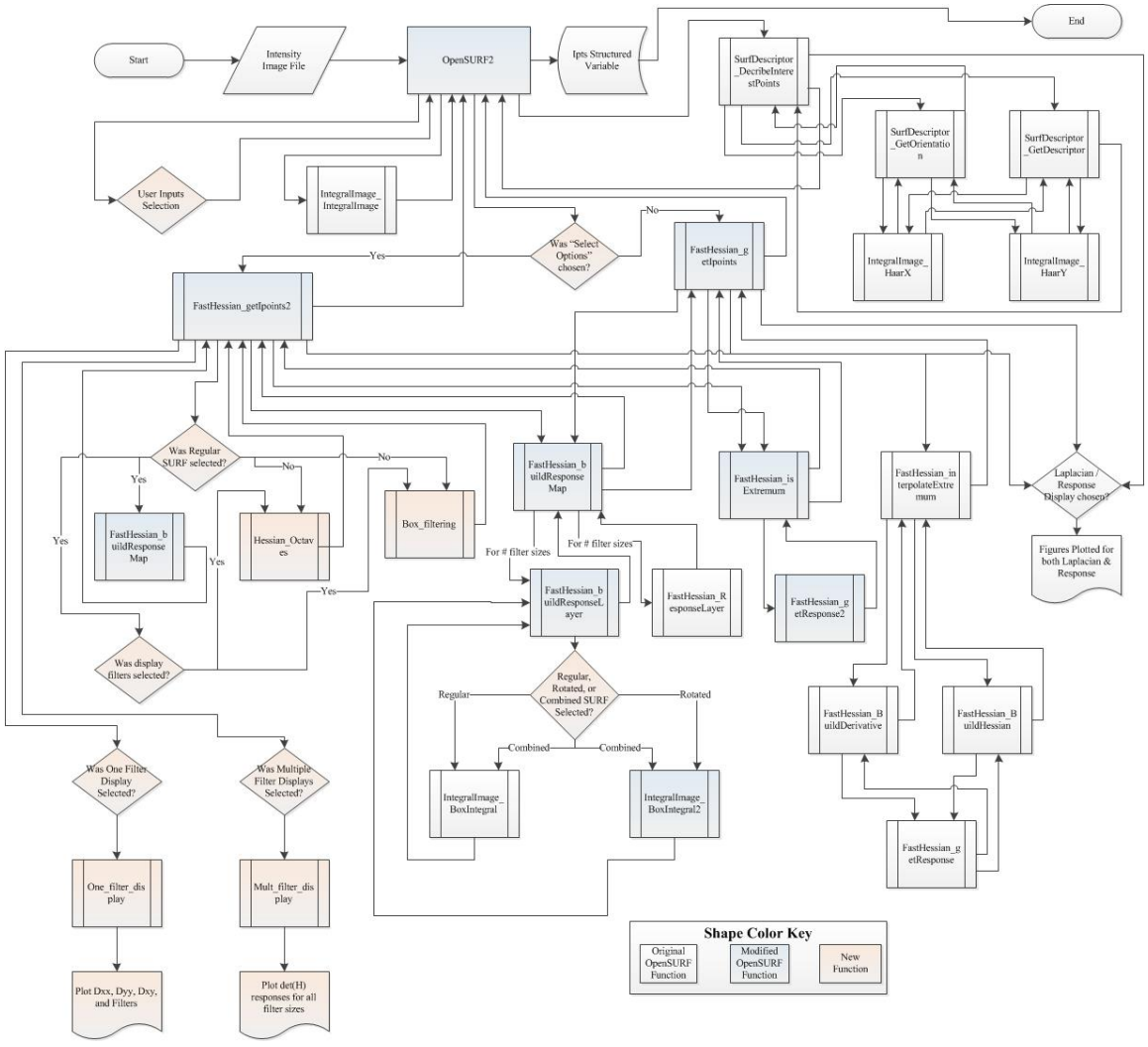

Figure 58. Modified OpenSURF algorithm flow diagram, after [25].

VI. APPLICATION TO AUTOMATED MARITIME VESSEL RECOGNITION

In order to correctly classify maritime vessels from images or video feeds, there needs to be a good combination of detector, descriptor, matching scheme, and finally filtering or organizing of results with a database, thresholding, or learning algorithm. Work by Rainey et al. was described in Section I.B. Additional work on integrating already existing detector, descriptor, and matching/learning algorithms for the purpose of ship detection (not recognition) was performed by Camp [51].

The research presented in this thesis focused on designing an improved feature detector by introducing rotated filters to the original SURF implementation. The resulting algorithm, called R-SURF, performed better than the regular SURF detector when facing JPEG compression and blurring degradations, which are two problems associated with image collection at sea. The R-SURF detector also performed well on viewpoint or affine invariance, which is crucial with all possible object or feature view angles. One weakness still present in the proposed R-SURF approach is image scale invariance. The rotated detector performs the best with smaller scale change on the tests conducted, which indicates that it may be well suited for the ranges that recognition would be performed between a maritime vessel and an image sensor on an aviation or surface platform; however, the proposed R-SURF approach is not suited for recognition of vessels at long distances in its current design.

Additionally, the computational load is a major concern for a detection-descriptor-matching process applied to continuous video feeds. For that reason, SURF or SIFT may not be best suited for the initial screening process. A faster algorithm with binary descriptors can be used as a screening for continuous video feeds such as the ORB or FREAK approaches briefly described in Sections II.A.4.d and II.A.4.f. Both algorithms were designed for low complexity monitoring of continuous video feeds, and may be useful tools prior to applying more sophisticated detectors such as SURF or R-SURF approaches for recognition.

Another obstacle to recognition is background clutter, especially in a port that may contain clutter that resembles the ship features. It is a very difficult problem to segment objects of interest from a similar background. If looking towards recognition and identification of an unknown maritime vessel, then knowing the characteristics a priori to differentiate the object from the background interest points is not usually an option. For the problem of clutter, getting the most correct information from each stage is crucial. Detectors must be repeatable and develop useful regions, descriptors must take those useful regions and accurately identify it with some invariance to image scene changes, and the matching and decision algorithms must be able to separate useful from non-useful points and group them to allow for successful recognition.

For the problem of automated maritime recognition, the most likely solution for aerial surveillance is 3D reconstruction of images and 3D image databases. 3D reconstruction is a computationally expensive solution that requires large databases but the only answer likely to result in quality recognition rates of maritime vessels in port scene. Bay, who developed the original SURF, did so with the goal of conducting 3D point and line correspondences for the purpose of reconstruction [18]. 3D reconstruction research can also be an extension of homography estimates derived from points of interest in for detector and descriptor evaluation of Mikolajczyz et al. [14], [15]. There is also a lot of other work in 3D reconstruction being accomplished by Google in Project Tango [52]. In the author's opinion, using interest point detectors and descriptors along with a matching strategy to reconstruct 3D images in a database is the path to high recognition rates with low false positives.

VII. CONCLUSION

In this research, the effectiveness of a scale space based computer vision algorithm, called SURF, has been investigated and a modification proposed to improve its performance. Particular attention was given to the detector, which need to identify a set of interest points to be characterized by the descriptor. This process needs to be robust to image transformations as outlined by Mikolajczyk et al. in their work on the performance of region detectors [15]. If a region detector does not provide an accurate region for orientation and descriptor generation, there is very little chance that a descriptor will be useful in follow-on matching schemes.

Inconsistencies in the existing SURF detector have been identified which make the algorithm less robust to image rotation and other transformations. It has been determined that this is a consequence of the Fast Hessian implementation, which tends to have inconsistencies with regards to detection in the face of different feature orientation, location within the image, and other deformation factors. The proposed filter modification improves upon most of these issues, with the possibility of further advances in varying the rotated design and kernel proportions.

In addition, we compared the performance of a number of SURF detectors under different perturbations, such as affine translations, varying illumination, JPEG compression, blurring, and scale transformations with rotation. The detectors tested include the regular SURF detector, a rotated SURF detector, a combination of equally weighted regular and rotated SURF detectors, and a combination of regular and rotated SURF detectors with each sample assigned a stochastic weighting. From these tests, the rotated detector clearly performs the best in the presence of blurring and JPEG compression of images. The regular detector appears to be the best performer in the presence of scale changes, but the rotated outperforms during smaller scale changes. Finally, the rotated detector performs the best for viewpoint or affine image changes on the textured scene, with all detectors performing poorly on the structured scene of graffiti. “

As became apparent, computer vision with dynamic image characteristics is a difficult problem that has many area that require additional work. The design requires significant effort in order to arrive at an algorithm and system that is robust to input image characters changes and has success rates that make it useful in practice. The goal for maritime application should be to get an input video or picture from any feed, cell phone to high-end detector to satellite image and have information pop up on an operator's screen with all the known data about the vessel of interest from a database. With the current progress occurring in computer vision, this is an attainable goal but requires effort to continue to design a system that is tailored to the specific application and the factors and associated problems. The results from this thesis present an improvement in the detector, which can be applied to improving the overall recognition process.

VIII. FUTURE WORK RECOMMENDATIONS

Computer vision and the associated fields are of great interest to the problem of automated recognition. There is work being accomplished in industry and the civilian sector that can be leveraged to apply to military applications, but there needs to be people dedicated to bringing that technology to a workable and cost-effective solutions that multiply force capabilities while reducing requirements for manning. In researching the topic, some areas have surfaced that are useful for future study and would greatly assist in the ultimate goal of maritime ship recognition.

A. DETECTOR IMPROVEMENT

In computer vision, the detection of points of interest locating features in an image, which was the focus of this thesis, is a wide-open research area in need of further development. The proposed modification to a 45° rotated Fast Hessian shows promise in multiple areas and should be further researched to see if different sizes can achieve better results. Also, the ability to match the original detector of SURF should be pursued.

Another possible approach to detector improvement is to use the fast integral image and box filtering techniques used by Bay et al. for the SURF algorithm and attempt to improve the detector with a slight loss in computational efficiency but a large gain in performance [6]. Pires et al. presented the idea of using the fast box filtering technique combined with more accurate representations of kernels like the Gaussian, Laplacian, and Hessian [22]. This work could be directly applied to modifying the SURF algorithm with a follow-on evaluation of performance against the original SURF and the R-SURF modification.

Additionally, many new, efficient algorithms use the FAST version of corner detection described in Section II.A.4.b. The BRISK [37], ORB [38], and FREAK [39] algorithms all use a version of the FAST detection method for interest points; while this technique may improve their computational efficiency, a corner detector is very different from the detectors used by SURF [6] and SIFT [16]. These differences merit

consideration when designing or using a detector for a particular application to ensure they are useful.

B. INTEREST POINT ORIENTATION ASSIGNMENT

Orientation assignment has a great impact on the performance of feature-based algorithms. The goal of this step is to properly align a detected feature so it can be described consistently under different image orientations. A consistent orientation results in a more accurate generation of descriptors for matching.

In Section II.A.2.b(1), a discussion on how the SURF algorithm determines orientation using x and y Haar wavelet responses was given. The SIFT algorithm uses a histogram of gradient magnitudes around an interest point as described in Section II.A.1. The ORB algorithm uses a vector defined by intensity moments around their interest point as described in Section II.A.4.d. Both the BRISK and FREAK algorithms sum the estimated local gradients over selected pairs of samples around the interest points, with FREAK using fewer sample points due to their claimed descriptor tolerance to orientation error [39].

As important as orientation seems to be, especially with the goal of image recognition, we were unable to find a standard test criteria that isolated orientation away from the detector or descriptor. Mikolajczyk et al. have developed a widely used test that isolates the detector [15] and descriptor [14], but the algorithm's orientation portion is only evaluated indirectly in the test that measures how well the descriptor performs. A standard testing method for testing orientation performance in needed. Otherwise, it is difficult to determine whether the descriptor or the orientation is at fault in the algorithm matching performance.

C. DESCRIPTOR IMPROVEMENT

The descriptor is an important step as it characterizes the object properties, both spatially and with regards to intensity. The descriptor has to be distinctive and able to distinguish between different objects while repeatable through image transformations. In Section II.A.2, it was shown that the detector region size greatly affects how the

descriptor is generated. This means that the detector and descriptor design should be accomplished as a system, where their performances are both individually and jointly evaluated.

We have shown how the SURF algorithm uses Haar wavelets that are rotated to correspond with the orientation to develop a descriptor in Section II.A.2.b(2). This descriptor captures intensity changes and can differentiate between low and high spatial frequency components. Additionally, it captures spatial information in the way the descriptor is generated [18]. One area for improvement is in performing empirical testing of different choices of wavelets that possibly might improve detection performance while maintaining computational efficiency.

Another possible area of research is a more targeted sampling pattern for SURF similar to that used for binary descriptor generation in the BRIEF [36], BRISK [37], or FREAK [39] algorithms. Additionally, the gradient histograms generated for sub-regions in SIFT described in Section II.A.1 performed well in testing as shown in Sections II.A.3 and 4. The SIFT descriptor can be improved by empirical testing as described for SURF. There is a balance of speed versus performance that needs to be obtained which is additionally affected by how fast the descriptor can be matched. As described in the summary and conclusion in Chapter VII, the binary descriptor may be better suited for filtering and monitoring, whereas the descriptors from SURF and SIFT may be more suited toward recognition. The end use must be considered when designing a descriptor to ensure it has the appropriate properties. The listed reasons signal that there is room for improvement in all descriptors that exist today, establishing a ripe area for research.

D. INTEREST POINT AND DESCRIPTOR MATCHING

We highlighted a few methods used for interest point matching in Section II.A.5. These range from individual descriptor matching methods such as basic threshold distance, nearest neighbor, and nearest neighbor distance ratio to methods that try to group points such as bag of visual words and interest point groups. There are also algorithms that attempt to identify particularly reliable and repeatable interest points such as RANSAC [45]. Additionally, work by Feineigle et al. that attempted to use spatial

information between interest points as a discriminatory method for interest points was covered in Section I.B. These are all methods deserving of additional research, especially with the application to automated ship recognition.

E. NEURAL NETWORKS

Neural networks have been shown to be very effective at discovering and recognizing patterns in data that human analysis may overlook. MATLAB provides a very user friendly interface for designing neural networks in their toolbox [53]. A topic for research could include how to organize the feature data, orientation, and spatial information to be applied to a neural network input layer and take advantage of pattern recognition capability. This technique can be applied to establish a framework for matching features or groups of features.

F. 3D IMAGE RECONSTRUCTION USING FEATURES

As described in Chapter VI, we believe 3D image reconstruction using correspondences between features can be a huge tool in achieving automated maritime recognition. This was the primary focus of Bay's work in [18]. Additionally, Google is working on 3D scene reconstructing in project Tango [52]. This type of reconstruction requires a sequence of images consistent with a video feed but provides depth and spatial context information for recognition that a 2D image does not capture. Additional work by Brown and Lowe with mapping a 2D plane to discover interest points along a planar surface also has promise in advancing the goal of 3D reconstruction [44].

G. BUILDING A MARITIME SPECIFIC IMAGE RECOGNITION DATABASE

We utilized a data set that was designed to test detectors for performance in detecting corresponding regions for transformed images [32]. This dataset has known homography between images that can be utilized in detector and descriptor performance testing [14], [15]. A lot of information in the development of homographic testing methods is available in the work of Mikolajczyk et al. [14], [15]. Hartley and Zisserman published a large body of work in multiple view geometry in computer vision that is a

good reference point for establishing base knowledge related to this subject [27]. Additionally, quality work was accomplished by Dubrofsky on homography estimation [54]. While it is easy to find images of different perspectives of maritime vessels for use in the detector-descriptor-matching sequence, developing a database of image with known homographies and a standard testing method allows more in-depth research of feature detectors and descriptors on maritime vessels. This database could serve a dual purpose of testing as well as the groundwork for a database with the purpose of 3D reconstruction.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. AFFINE COVARIANT DATA SET TEST RESULTS

A. BARK AND LEAVES TEST SET RESULTS

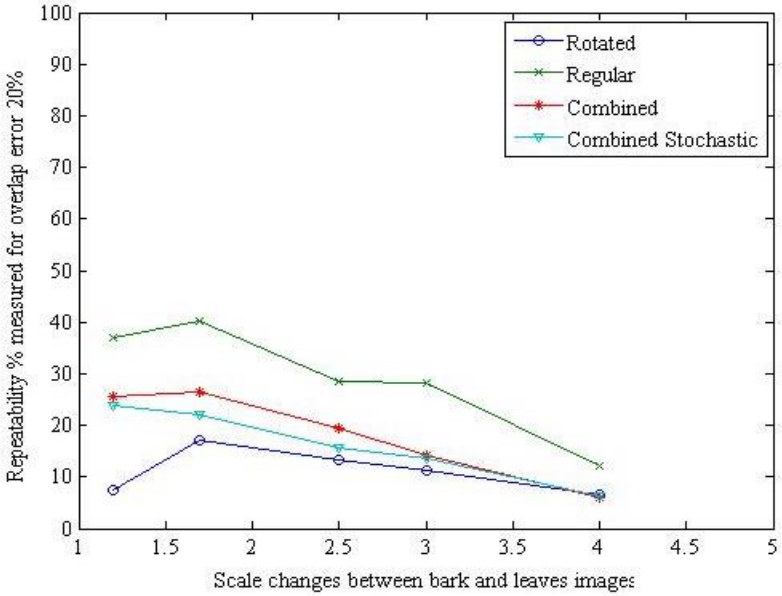

Figure 59. Repeatability versus scale changes for tested detectors on bark and leaves test set with ε_o threshold value equal to 0.2.

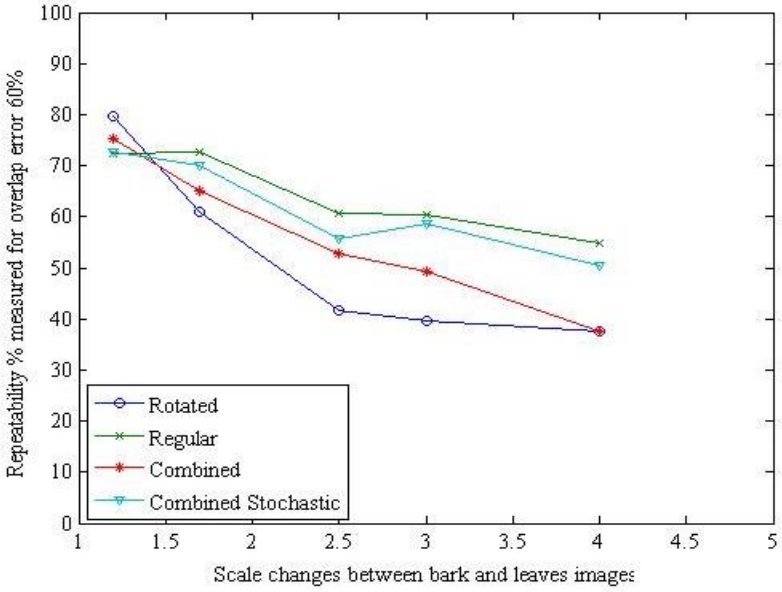

Figure 60. Repeatability versus scale changes for tested detectors on bark and leaves test set with ε_o threshold value equal to 0.6.

B. BIKE TEST SET RESULTS

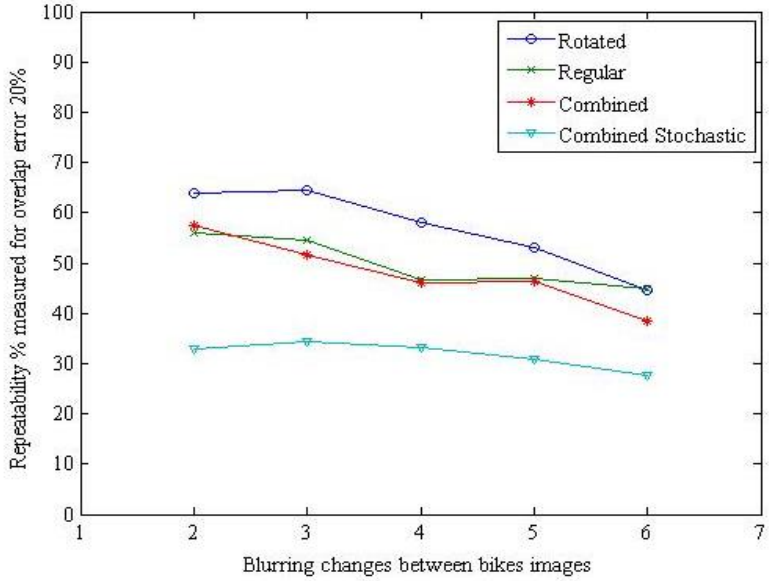

Figure 61. Repeatability versus blurring changes for tested detectors on bikes test set with ε_o threshold value equal to 0.2.

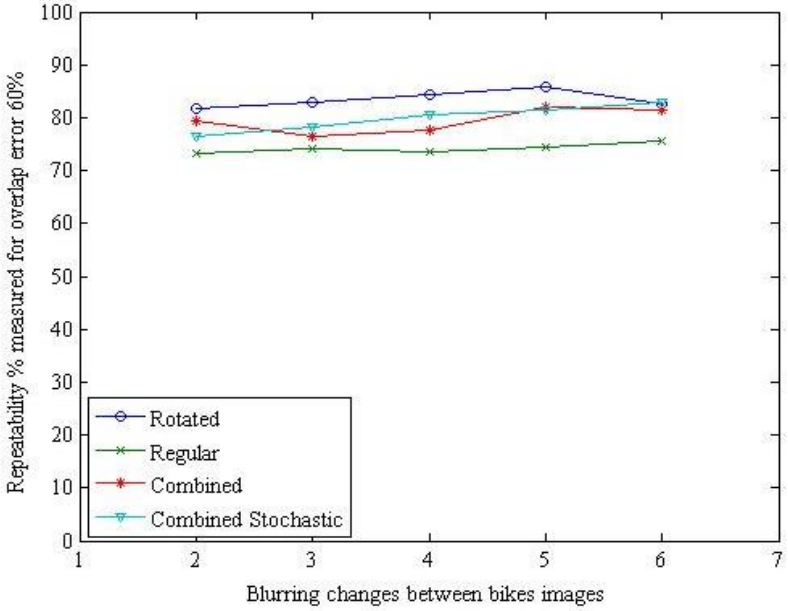

Figure 62. Repeatability versus blurring changes for tested detectors on bikes test set with ε_o threshold value equal to 0.6.

C. BOAT TEST SET RESULTS

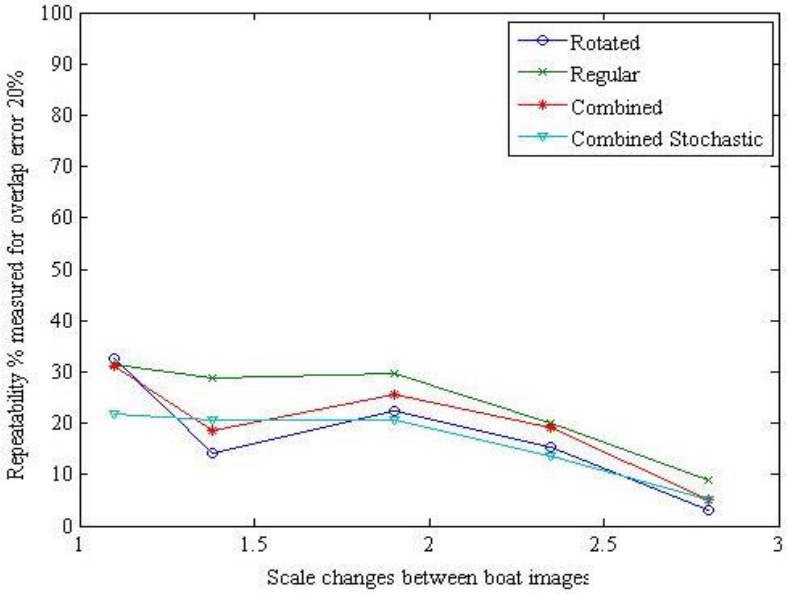

Figure 63. Repeatability versus scale changes for tested detectors on boat test set with ε_o threshold value equal to 0.2.

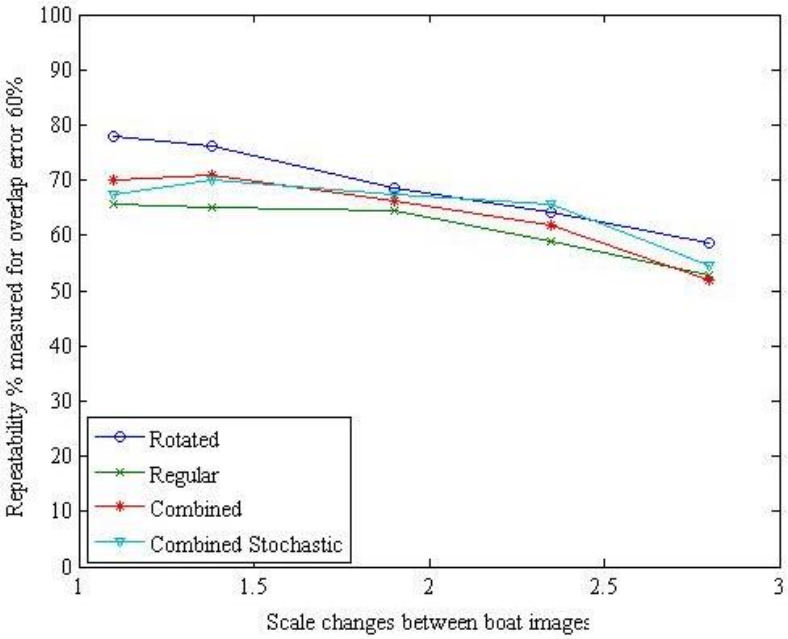

Figure 64. Repeatability versus scale changes for tested detectors on boat test set with ε_o threshold value equal to 0.6.

D. GRAFFITI TEST SET RESULTS

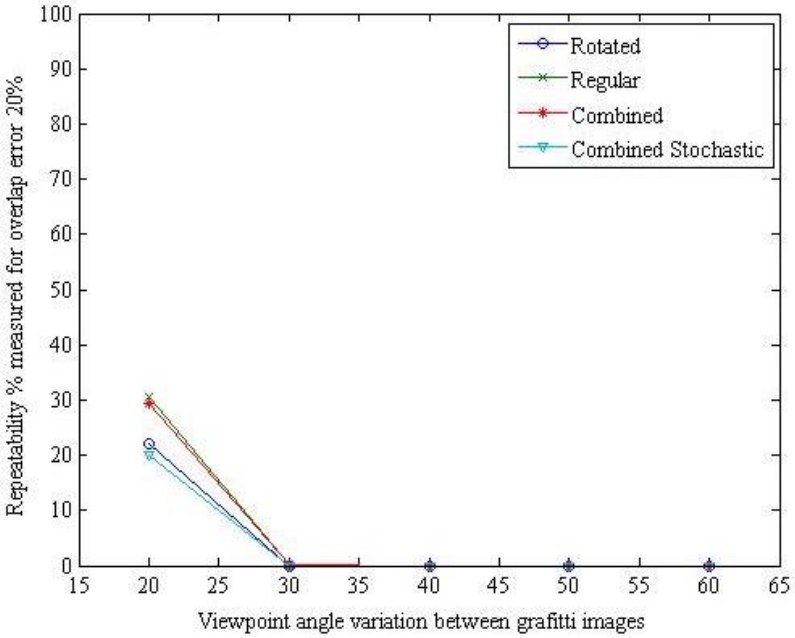

Figure 65. Repeatability versus viewpoint changes for tested detectors on graffiti test set with ε_o threshold value equal to 0.2.

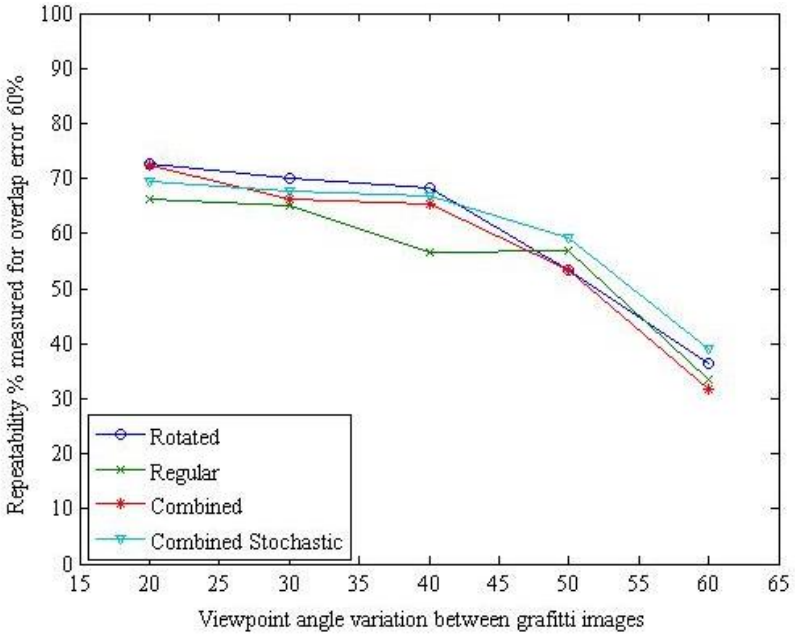

Figure 66. Repeatability versus viewpoint changes for tested detectors on graffiti test set with ε_o threshold value equal to 0.6.

E. CARS AND BUILDING TEST SET RESULTS

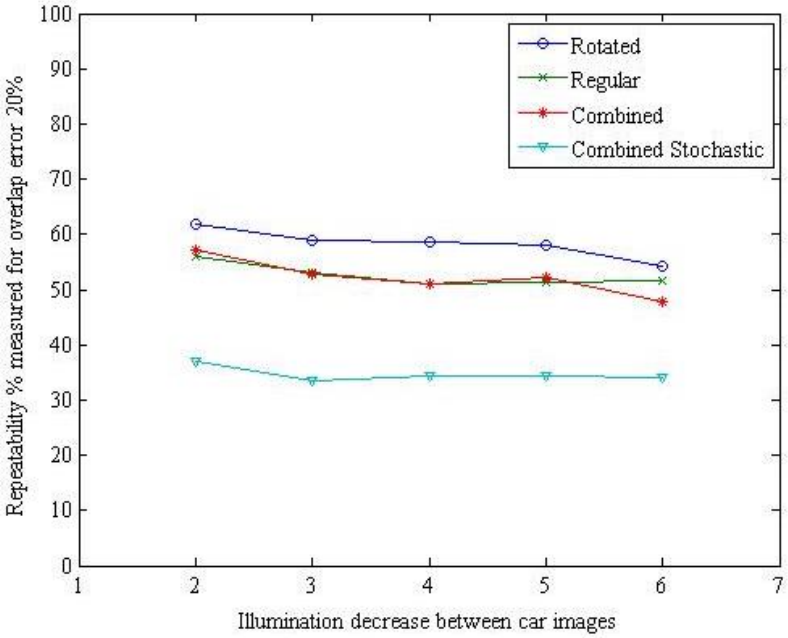

Figure 67. Repeatability versus illumination changes for tested detectors on cars and building test set with ε_o threshold value equal to 0.2.

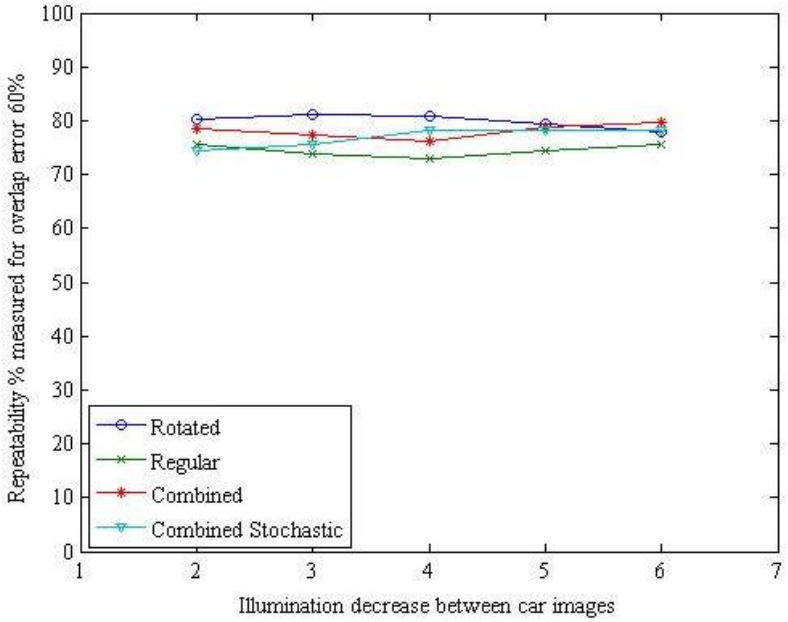

Figure 68. Repeatability versus illumination changes for tested detectors on cars and building test set with ε_o threshold value equal to 0.6.

F. TREES TEST SET RESULTS

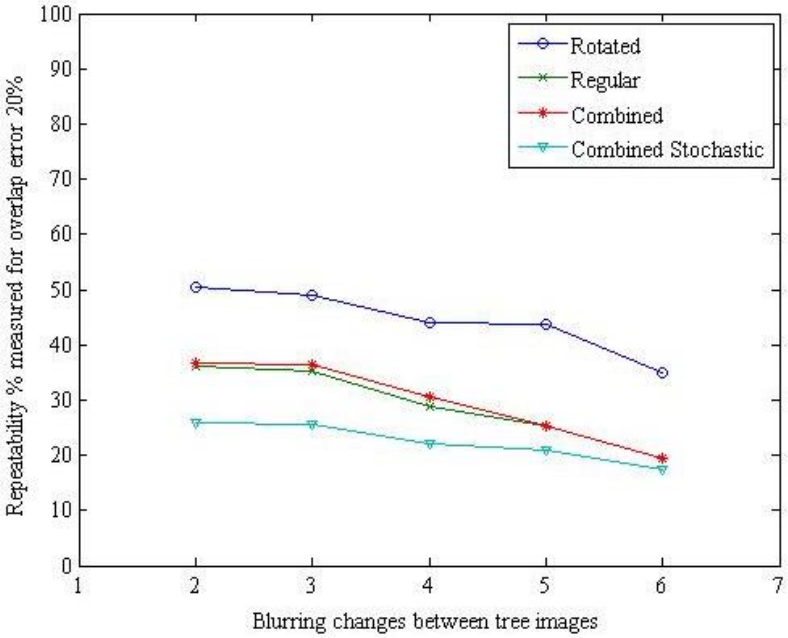

Figure 69. Repeatability versus blurring changes for tested detectors on trees test set with ε_o threshold value equal to 0.2.

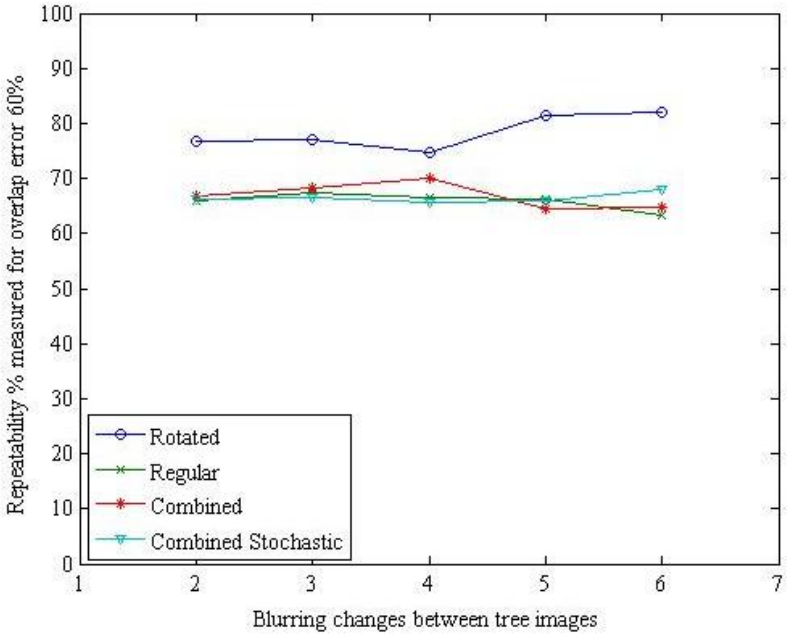

Figure 70. Repeatability versus blurring changes for tested detectors on trees test set with ε_o threshold value equal to 0.6.

G. BUILDING TEST SET RESULTS

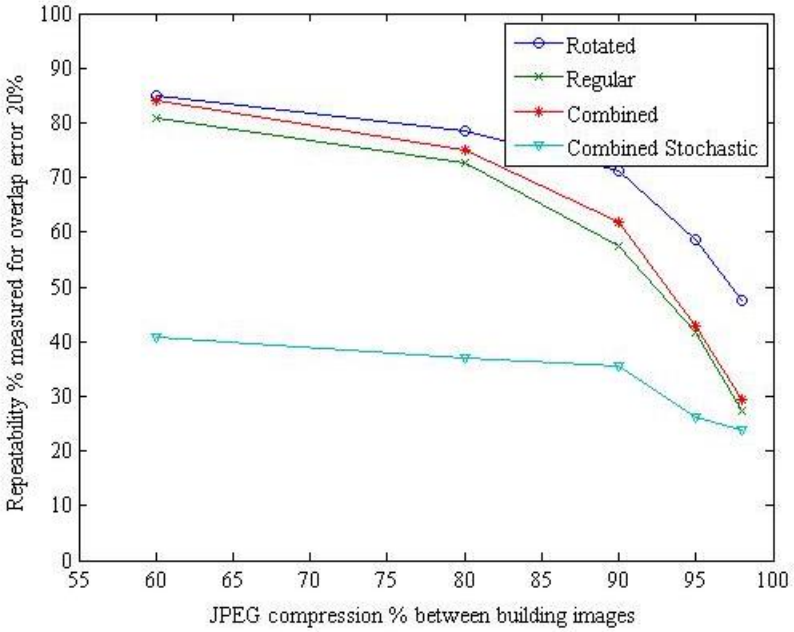

Figure 71. Repeatability versus JPEG compression changes for tested detectors on building test set with ε_o threshold value equal to 0.2.

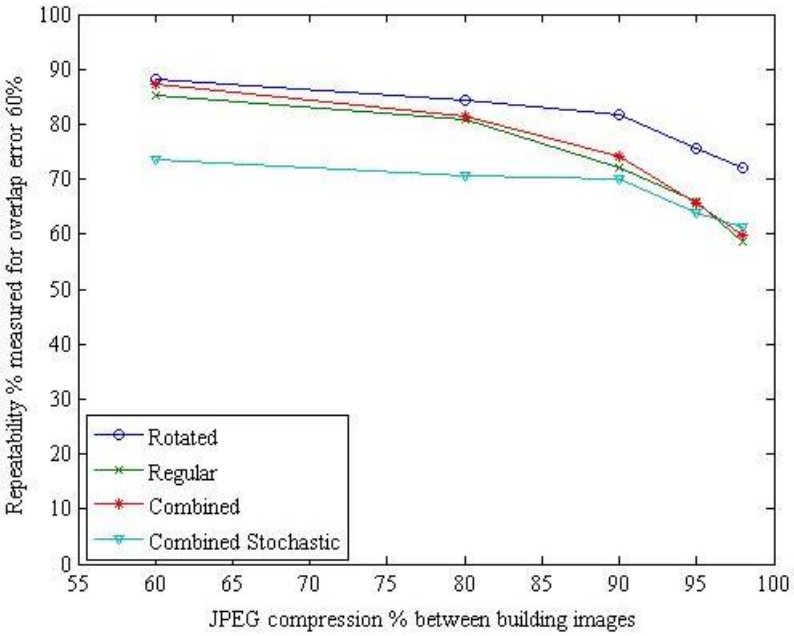

Figure 72. Repeatability versus JPEG compression changes for tested detectors on building test set with ε_o threshold value equal to 0.6.

H. BRICK WALL TEST SET RESULTS

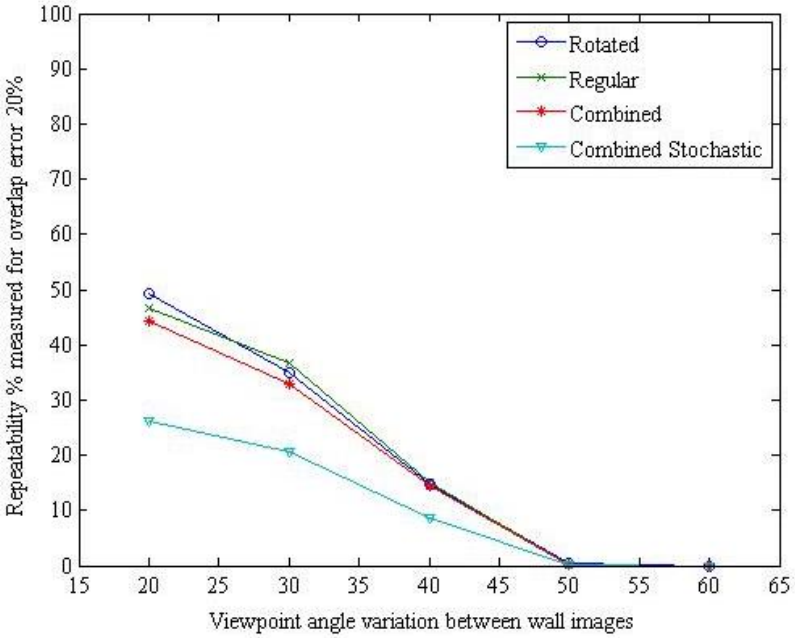

Figure 73. Repeatability versus viewpoint changes for tested detectors on brick wall test set with ε_o threshold value equal to 0.2.

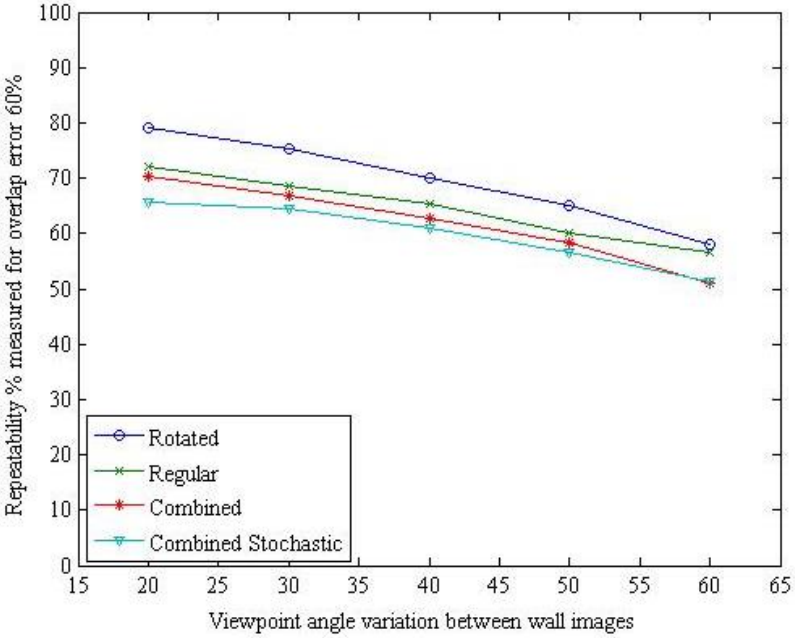

Figure 74. Repeatability versus viewpoint changes for tested detectors on brick wall test set with ε_o threshold value equal to 0.6.

SUPPLEMENTAL MATLAB CODE

The OpenSURF coding modifications to the original OpenSURF version 1c file from the Mathworks website is available as an electronic document. The MATLAB files and test files are contained in a compressed file named 14Jun_Jurgensen_Sean_Supplemental_MATLABCode.zip. The files contain modifications to the original OpenSURF program and testing programs and images obtained from open-source websites [25], [32] ,[50]. Contact the Dudley Knox Library at the Naval Postgraduate School to obtain a copy at (831) 656-2947 or circdesk@nps.edu. As requested in the OpenSURF program on the Mathworks website [25], the following statement is provided:

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] United States Geological Survey, "How much water is there on, in, and above the Earth?," 19 Mar. 2014. [Online]. Available: http://water.usgs.gov/edu/earthhowmuch.htm

[2] United States Navy, "RQ-8A and MQ-8B Fire Scout unmanned aerial vehicle (UAV)," United States Navy Fact File, 18 Feb. 2009. [Online]. Available: http://www.navy.mil/navydata/fact_display.asp?cid=1100&tid=2150&ct=1

[3] United States Navy, "RQ-2A Pioneer unmanned aerial vehicle (UAV)," United State Navy Fact File, 9 Sept. 2013. [Online]. Available: http://www.navy.mil/navydata/fact_display.asp?cid=1100&tid=2100&ct=1

[4] United States Navy, "MINIROVS-remotely operated vehicle," United States Navy Fact File, 22 Nov. 2013. [Online]. Available: http://www.navy.mil/navydata/fact_display.asp?cid=4300&tid=93&ct=4

[5] International Commission on Illumination, "CIE home," 1 Jan. 2014. [Online]. Available: http://cie.co.at/

[6] H. Bay, Tuytelaars, T. Tuytelaars and L. V. Gool, "SURF: speeded-up robust features," in *European Conf. on Computer Vision*, Graz, Austria, 2006.

[7] D. Gossow, P. Decker and P. Dietrich, "An evaluation of open source SURF implementations," in *Robot Soccer World Cup XIV*, Springer-Verlag, Berlin, Heidelberg, 2011.

[8] L. Juan and O. Gwon, "A comparison of SIFT, PCA-SIFT and SURF," *Int. Journal of Image Processing*, vol. 3, no. 4, pp. 143–152, 2009.

[9] Y. N. Khan, B. McCane and G. Wyyill, "SIFT and SURF performance evaluation against various image deformations on benchmark dataset," in *Internataional Conf. on Digital Image Computing Techniques and Applications*, Noosa, QLD, Australia, 2011.

[10] J. Bauer, N. Sunderhauf and P. Protzel, "Comparing several implementations of two recently published feature detectors," in *Proc. of the Int. Conf. on Intelligent and Autonomous Systems*, Toulouse, France, 2007.

[11] K. Rainey, S. Parameswaran, J. Harguess and J. Stastny, "Vessel classification in overhead satellite imagery using," in *SPIE 8499 Application of Digital Image Processing XXXV 84992F,* San Diego, CA, 2012.

[12] P. A. Feineigle, D. D. Morris and F. D. Snyder, "Ship recognition using optical imagery for harbor survelliance," in *Association for Unmanned Vehicle Systems International*, Washington, DC, 2007.

[13] R. Szeliski, *Computer Vision: Algorithms and Applications*, London: Springer, 2010.

[14] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Patter Analysis and Machine Intelligence,* vol. 27, no. 10, pp. 1615-1630, 2005.

[15] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir and L. Van Gool, "A comparison of affine region detectors," *Int. Journal of Computer Vision,* vol. 65, no. 1/2, pp. 43-72, 2005.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints.," *Int. Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[17] H. Bay, E. A and L. Van Gool, "SURF: speeded-up robust features," *Computer Vision and Image Understanding (CVIU),* vol. 110, no. 3, pp. 346–359, 2008.

[18] H. Bay, "From wide-baseline point and line correspondence to 3D," Ph.D. dissertation, Eidgenössische Technische Hochschule Zurich, Zurich, Switzerland, 2006.

[19] T. Lindenberg, "Scale space theory: A basic tool for analyzing structures and different scales," *Journal of Applied Sciences,* vol. 21, no. 2, pp. 225–270, 1994.

[20] D. G. Lowe, "Object recognition from local scale-invariant features," in *Int. Conf. on Computer Vision*, Corfu, Greece, 1999.

[21] P. Viola and M. Jones, "Robust real-time face detection," *Int. Journal of Computer Vision,* vol. 57, no. 2, pp. 137–154, 2004.

[22] B. Pieres, K. Signgh and J. Moura, "Approximating image filters with box filters," in *2000 18th IEEE Conf. on Image Processing (ICIP),* Vancouver, British Columbia, 2011.

[23] K. Sung and R. Casper, "Application of convolution in image processing with MATLAB," University of Washington, Seattle, WA, 2013.

[24] C. Evans, “Notes on the OpenSURF library,” The OpenSURF Computer Vision Library, Stevenage, UK, 2009.

[25] D. Jan Kroon, “OpenSURF (including image warp),” 06 Oct.2010. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/28300-opensurf--including-image-warp-

[26] A. Ben-Menahem, Historical Encyclopedia of Natural and Mathematical Sciences, vol. 1, New York: Springer-Verlag Berlin Heidelberg, 2009, p. 1972.

[27] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge, UK: Cambridge University Press, 2003.

[28] D. Rákos, “Rastergrid blogosphere,” 7 Sep. 2010. [Online]. Available: http://rastergrid.com/blog/2010/09/efficient-gaussian-blur-with-linear-sampling/

[29] A. Neubeck and L. Van Gool, “Efficienient non-maximum suppression,” in *18th Concference on Pattern Recogniation*, Zurich, Switzerland, 2006.

[30] C. Papageorgiou, C. Oren and T. Poggio, “A general framework for object detection,” in *Sixth Int. Conf. on Computer Vision*, Washington, DC, 1998.

[31] Swiss Federal Institute of Technology, “SURF feature detector sofware Licence,” *Computer Vision Laboratory*, p. 1, 7 Nov. 2013.

[32] University of Oxford, “Visual Geometry Group,” 19 Sep. 2004. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/data-aff.html

[33] K. Yan and R. Sukthankar, “PCA-SIFT: A more distinctive represenation for local image descriptors,” in *Proceeding Conf. on Computer Vision and Pattern Recognition*, Washington, DC, 2004.

[34] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Fourth ALvey Vision Conf.*, Manchester, England, 2012.

[35] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Proc. of the European Conf. on Computer Vision*, Graz, Austria, 2006.

[36] M. Calonder, V. Lepetit, C. Strecha and P. Fua, “BRIEF: binary robust independent elementary features,” in *Proc. of the European Conf. on Computer Vision*, Crete, Greece, 2010.

[37] S. Leutenegger, M. Chli and R. Siegward, "BRISK: binary robust invariant scalable keypoints," in *IEEE Proc. Int. Conf. on Computer Vision*, Barcelona, Spain, 2011.

[38] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *IEEE Int. Conf. on Computer Vision*, Barcelona, Spain, 2011.

[39] A. Alahi, R. Ortiz and P. Vandergheynst, "FREAK: fast retina keypoint," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Providence, RI, 2012.

[40] M. Muja and D. G. Lowe, "Fast approximation nearest neighbor with automatic algorithm configuration," in *VISAPP Int. Conf. on Computer Vision Theory and Application*, Lisboa, Portugal, 2009.

[41] J. Sivic and A. Zisserman, "Efficient visual search of videos cast as text retrieval," *IEEE Transactions on Pattern Analysis and Machine Learning*, vol. 31, no. 4, pp. 591-606, 2009.

[42] Y. Lei, L. Jing and X. Changsheng, "Descriptive local feature groups for image classification," in *18th IEEE Int. Conf. on Image Processing*, Brussels, Belguim, 2011.

[43] E. S. Ng and N. G. Kingsbury, "Matching of interest point groups with pairwise spatial constraints," in *Proc. 17th IEEE Int. Conf. on Image Processing*, Hong Kong, 2010.

[44] M. Brown and D. Lowe, "Invariant features from interest point groups," in *British Matching Vision Conf.*, Cardiff, UK, 2002.

[45] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the Association for Computing Machinery*, vol. 24, no. 6, pp. 381-395, 1 Jun.1981.

[46] C. Song, H. Zhao, W. Jing and Z. Hongbo, "Robust video stabilization based on particle filtering with weighted feature points," *IEEE Transactions on Computer Electronics*, vol. 58, no. 2, pp. 570577, 2012.

[47] Open Source Initiative, "The BSD 3-clause license," [Online]. Available: http://opensource.org/licenses/BSD-3-Clause

[48] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," *IEEE Proc. from the Int. Conf. on Image Processing*, pp. 900–903, 2002.

[49] K. G. Derpanis, “Integral image-based representations,” Department of Computer Science and Engineering York University, Toronto, Canada, 2007.

[50] Visual Geometry Group, Katholieke Universiteit Leuven, Inria Rhone-Alpes, Center for Machine Perception, “Comparison of region descriptor,” 16 Mar. 2006. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/research/affine/desc_evaluation.html#code

[51] D. Camp, “Evaluation of object detection algorithms for ship detection in the visible spectrum” M.S, thesis, Dept. of Elect. and Comp. Eng., Naval Postgraduate School, Monterey, CA, 2013.

[52] Google, “Project Tango,” [Online]. Available: https://www.google.com/atap/projecttango/

[53] Mathworks, “Neural Network Toolbox,” 2004. [Online]. Available: http://www.mathworks.com/products/neural-network/description1.html

[54] E. Dubrofsky, “Homography Estimation,” M.S. thesis, Dept. of Comp. Science, Carleton University, Ottawa, Ontario, Canada, 2007.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California